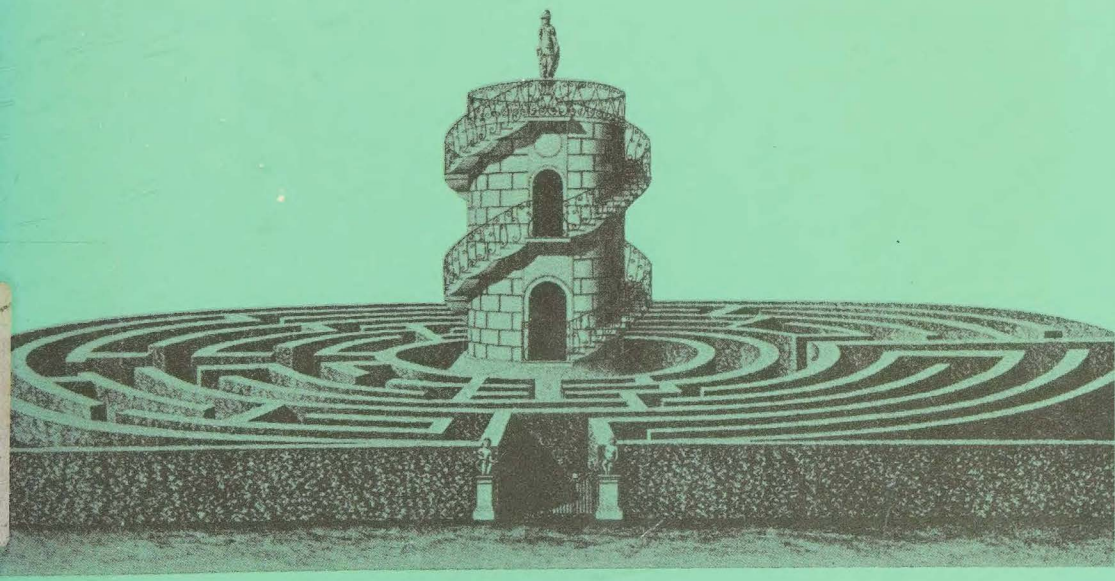
Architecture des ordinateurs

tome 2

Systèmes d'exploitation et extensions du matériel

J.-A. MONTAGNON

Préface de C. KAISER



MASSON

ARCHITECTURE DES ORDINATEURS

Tome 2

CHEZ LE MÊME ÉDITEUR

Du même auteur :

ARCHITECTURE DES ORDINATEURS.
Tome 1. — Le sous-système central, par J.-A. MONTAGNON et E. PICHAT. 1986, 232 pages.

Autres ouvrages :

PRINCIPES DE FONCTIONNEMENT DES ORDINATEURS, par R. DOWSING et F. WOODHAMS. Traduit de l'anglais par E. David. *Collection Manuels Informatiques Masson.* 1987, 160 pages.

LES SYSTÈMES D'EXPLOITATION. Structure et concepts fondamentaux, par C. LHERMITTE. *Collection Manuels Informatiques Masson.* 1985, 188 pages.

SYSTÈMES D'EXPLOITATION : CONCEPTION ET FONCTIONNEMENT, par D.W. BARRON. Traduit de l'anglais par A. Bernard. *Collection Manuels Informatiques Masson.* 1986, 128 pages.

UNIX, SYSTÈME ET ENVIRONNEMENT, par A.-B. FONTAINE et Ph. HAMMÈS. *Collection Manuels Informatiques Masson.* 1986, 3e tirage, 228 pages.

MULTICS. Guide de l'usager, par J. BERSTEL et J.-F. PERROT. *Collection Manuels Informatiques Masson.* 1986, 280 pages.

PROCESSUS CONCURRENTS. Introduction à la programmation parallèle, par M. BEN-ARI. Traduit de l'anglais par A. Bernard. *Collection Manuels Informatiques Masson.* 1986, 176 pages.

LE MICROPROCESSEUR 16 BITS-8086/8088. Matériel, logiciel, système d'exploitation, par A.-B. FONTAINE. *Collection Manuels Informatiques Masson.* 1984, 2e édition étendue aux 1 86 et 2 86, 240 pages.

LES MICROPROCESSEURS 80286/80386. Nouvelles architectures PC, par A.-B. FONTAINE et F. BARRAND. *Collection Manuels Informatiques Masson.* 1987, 320 pages.

LE MICROPROCESSEUR MC 68000. Contrôle et mise au point des composants associés, par J.W. COFFRON. Traduit de l'anglais par M. Boulafa. *Collection Manuels Informatiques Masson.* 1985, 216 pages.

ASSEMBLEUR SUR IBM PC, par D.J. BRADLEY. Traduit de l'américain par L. de Pardieu. 1986, 384 pages.

TRANSMISSION ET RÉSEAUX LOCAUX. Architecture IEEE 802, par P. MILLET. *Collection Méthodes Informatiques et Pratique des Systèmes.* 1987, 232 pages.

TESTS DES CIRCUITS NUMÉRIQUES. Aide à la programmation, par J.-C. COUCHOT. *Collection Méthode + Programmes.* 1986, 128 pages.

ARCHITECTURE DES ORDINATEURS

Tome 2

SYSTÈMES D'EXPLOITATION ET EXTENSIONS DU MATÉRIEL

J.-A. MONTAGNON

Ingénieur ESE
Docteur ingénieur IMAG
Consultant en réseaux d'entreprise

Préface de C. KAISER

MASSON
Paris New York Barcelone Milan
Mexico São Paulo
1987

Illustration de la couverture :
Labyrinthe de la Villa Pisani à Stra.
D.R.

Tous droits de traduction, d'adaptation et de reproduction par tous procédés, réservés pour tous pays.

La loi du 11 mars 1957 n'autorisant, aux termes des alinéas 2 et 3 de l'article 41, d'une part, que les « copies » ou « reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective », et d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause, est illicite » (alinéa 1er de l'article 40).

Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles 425 et suivants du Code pénal.

© *Masson, Paris, 1987*

ISBN : 2-225-81184-9

MASSON S.A. 120, bd Saint-Germain, 75280 Paris Cedex 06
MASSON S.A. Balmes 151, 08008 Barcelona
MASSON ITALIA EDITORI S.p.A. Via Giovanni Pascoli 55, 20133 Milano
MASSON EDITORES Dakota 383, Colonia Napoles, 03810 Mexico D.F.
EDITORA MASSON DO BRASIL Ltda Rua Borges Lagoa 1044, CEP 04038 São Paulo S.P.

préface

Depuis plus de vingt ans, les systèmes d'exploitation passionnent et intriguent les informaticiens tant par leur complexité et leur puissance que par leur taille et leur difficulté de mise en oeuvre.

Le système d'exploitation est l'habit de lumière des ordinateurs, qui transforme une machine rétive en un pur-sang obéissant au doigt et à l'oeil des utilisateurs, ... ou qui devrait faire ce miracle. Quelquefois aussi des machines prometteuses ont été perdues par de mauvais systèmes.

Le système d'exploitation est donc cette touche intermédiaire entre la machine et les divers utilisateurs par laquelle on tente de réaliser, pour un type d'exploitation donné, le meilleur compromis entre les performances globales du matériel, le confort d'accès des utilisateurs et la prise en compte du plus grand nombre de fonctions possibles.

Je classerais les auteurs de livres sur les systèmes d'exploitation en deux catégories.

Il y a ceux qui ont construit des systèmes, ou plutôt des noyaux de système, ce sont des universitaires ou des industriels constructeurs; leur souci est de présenter les choix fondamentaux, d'expliquer pourquoi ils éliminent certains dispositifs, et comment ils ont hiérarchisé les mécanismes du système. Ils sont amenés pour cela à distinguer trois grandes classes du système:

- la conduite et la synchronisation des activités qui concourent au fonctionnement du service;
- la répartition dynamique et l'utilisation des ressources communes;
- le contrôle et l'accès aux informations engrangées dans le système.

Cela leur permet de rattacher des systèmes à chacune de ces classes. Et puis tous les autres problèmes sont, il faut bien le reconnaître, traités bien plus superficiellement.

Il y a ceux qui utilisent les systèmes et qui doivent savoir expliquer tout ce qui se passe dans un système d'exploitation sans avoir le droit d'éliminer quoi que ce soit puisque, par définition, il faut tout dire. Ces auteurs choisissent en général de présenter un seul système d'exploitation, ce qui est déjà beaucoup lorsqu'il s'agit

d'un grand système d'exploitation professionnel et ce qui a l'avantage de limiter le domaine à aborder.

L'ouvrage de J.-A. Montagnon se rattache à cette deuxième catégorie avec plus d'ambition encore car il prend ses exemples dans plusieurs systèmes industriels.

Voici un livre que peu de gens pourraient écrire car il suppose une connaissance énorme sur les systèmes d'exploitation. Voici un livre dont je recommande la lecture en parallèle avec un ouvrage sur les principes des systèmes d'exploitation. En effet, le livre de J.-A. Montagnon présente une multitude d'exemples en montrant bien les ressemblances de fonction et de structure d'un constructeur à l'autre, même quand les noms utilisés diffèrent pour des raisons commerciales. Loin de fournir un simple catalogue, l'auteur s'est livré à un véritable effort de synthèse. Il aborde sans barguigner tous les aspects pragmatiques qui intéressent l'utilisateur comme les services, les versions, la facturation, l'organisation du service informatique, la planification des travaux, la gestion des fichiers, la sécurité, la protection, la génération, l'installation, tout autant que la gestion des processus, des entrées-sorties ou de la mémoire virtuelle.

Il est évident qu'on peut lui faire quelques reproches. Il est inhérent à la tentation encyclopédique de ne pouvoir tout dire et tout classer dans une discipline en évolution rapide. J'aurais aimé une présentation détaillée de l'arborescence des fichiers de Multics et d'Unix, un exposé du langage de commande d'Unix, une description des langages de commandes avec icônes, un aperçu des systèmes tolérant les pannes. Cet ouvrage prend surtout ses exemples dans la galaxie IBM et donc dans des systèmes d'exploitation qui privilégient le partage de ressources.

Une autre raison, très simple, pour laquelle je recommande vivement cet ouvrage est qu'il est écrit en bon français. Le jargon professionnel et l'inflation des termes anglo-saxons se limitent aux noms sans envahir toute la phrase comme il arrive trop souvent. Le texte de cet ouvrage est en français et il faut saluer cet effort! Les mots du jargon sont encadrés entre parenthèses ou guillemets, ils sont mis en relation avec d'autres mots du jargon qui expriment la même idée et souvent un terme français est proposé en traduction. J'y ai ainsi trouvé avec plaisir la traduction de "overhead" en déperdition et celle de "thrashing" en emballement.

Je pense que cet ouvrage, qui s'appuie sur une longue expérience d'enseignement et une grande pratique des systèmes, et qui, à ma connaissance, est unique dans le domaine, devrait toucher de très nombreux lecteurs. A ceux-ci je m'adresse en paraphrasant F. Nietzsche:

Ami lecteur, bonne mâchoire, bon estomac
C'est ce que je te souhaite:
Quand tu auras digéré ce livre
Tu t'entendras certainement avec son auteur.

C. KAISER
Professeur au Conservatoire National des Arts et Métiers.

table des matières

Introduction 11
1 - Définitions et généralités 15
1.1 - Définition générale 15
1.2 - Les utilisateurs du SE 15
1.3 - Structure générale du logiciel 16
1.4 - Les versions d'un logiciel 18
1.5 - Portabilité, pérennité 19
1.6 - Facturation 19
1.7 - Les fournisseurs 20
1.8 - Importance du logiciel 20
1.9 - Langages d'écriture des systèmes d'exploitation 22
1.10 - Conclusion 22
2 - Fonctions du système d'exploitation 23
2.1 - Les fonctions générales 23
2.2 - La gestion des ressources 24
2.2.1 - Classification des ressources 24
2.2.2 - Mécanismes d'attribution des ressources 24
2.3 - Gestion du système informatique 25
2.3.1 - Organisation d'un service informatique 25
2.3.2 - Fonctions de gestion du système 27
3 - Les types de système d'exploitation 28
3.1 - Typologie 28
3.2 - SE orientés vers le mode différé 29
3.3 - SE orienté vers le mode dialogué 31
3.4 - Les SE orientés vers le temps réel 33
3.5 - SE et tailles des ordinateurs 36
3.6 - SE et architecture externe 36
4 - Structure générale d'un système d'exploitation 37
4.1 - Les trois phases d'une requête 37
4.2 - Le système d'entrée des requêtes 38
4.2.1 - Lecture des requêtes 38
4.2.2 - Analyse 39
4.2.3 - Planification 40
4.2.4 - Cas des SE interactifs 43
4.2.5 - Cas des multisystèmes 45
4.3 - Allocation des ressources 46
4.3.1 - Ordre d'allocation 47
4.3.2 - Allocations statique et dynamique 48
4.3.3 - Problèmes d'interblocage 49
4.3.4 - Désallocation 50
4.3.5 - Types, classes de ressources 51
4.4 - Sous-système de sortie 52
4.5 - Gestion des travaux, enchaînement, noyau 53
4.6 - Machines multi-SE 55
4.7 - SE multiprocesseur/système 58

5 - Allocation des ressources 61
5.1 - Notion de processus 61
5.1.1 - Création des processus 62
5.1.2 - Etats d'un processus 63
5.1.3 - Relations entre processus 64
5.2 - Allocation d'espace sur disque 64
5.2.1 - Allocations statique et dynamique 65
5.2.2 - Allocation contiguë 66
5.2.3 - Gestion de la place sur disque 68
5.2.4 - Organisation de l'espace d'un fichier 69
5.2.5 - Gestion de l'espace défectueux 71
5.2.6 - Choix du disque 71
5.2.7 - Sécurité, confidentialité 72
5.3 - Allocation de l'accès aux disques 73
5.4 - Allocation de la mémoire centrale 74
5.5 - Allocation du processeur 75
5.5.1 - Priorités fixes 75
5.5.2 - Tranches de temps 76
5.5.3 - Priorités fixes et tranches de temps 76
5.5.4 - Priorités dynamiques 78
5.5.5 - Cas des multiprocesseurs 80
5.6 - Allocation des canaux 80
5.7 - Le noyau vu de l'extérieur 81
5.8 - Résumé, structure des SE 82
6 - Gestion des entrées/sorties **83**
6.1 - Les différents types d'entrées/sorties 83
6.2 - Les entrées/sorties physiques 84
6.3 - Entrées/sorties synchrones et asynchrones 87
6.4 - Entrées/sorties logiques 89
6.4.1 - Modes "move" et "locate" 92
6.4.2 - Tampons multiples (multiple buffering) 94
6.4.3 - Tampons multiples en rafale 95
6.5 - Gestion des entrées/sorties lentes 96
6.5.1 - SYSIN, SYSOUT, spoule 96
6.5.2 - Cas de processeurs spécialisés 97
6.6 - Entrées/sorties virtuelles 98
6.7 - Cas des multiprocesseurs 100
6.8 - Gestion des interruptions (GINT) 100
7 - Gestion des fichiers **102**
7.1 - Les différents niveaux 102
7.2 - Les fonctions du SGF 103
7.3 - Structure du répertoire 104
7.3.1 - Sous-répertoires 105
7.3.2 - Répertoire courant 106
7.3.3 - Relation utilisateur-répertoire 106
7.3.4 - Fichier multirépertoire 107
7.3.5 - Extension au réseau 107
7.4 - Types de volume et de fichier, relations 107
7.5 - Gestion des périphériques 109
7.6 - Protection de l'accès 110
7.7 - Accès concurrent 113
7.8 - Cas des multisystèmes 115
7.9 - Gestion des versions 115
7.10 - Domaines couverts par les SGF 115
7.11 - SE multi-SGF 116
7.12 - Taille des fichiers 117
8 - Gestion des données **118**
8.1 - Les méthodes d'accès 118
8.2 - Les types d'organisation 119
8.2.1 - Organisations des fichiers classiques 119
8.2.2 - Organisations liées aux bases de données 124
8.3 - Les modes d'accès 127
8.4 - Relations entre bloc et articles 128
8.5 - Compression des données 129

9 - Communication et synchronisation entre processus 130
9.1 - Communication avec le SE 130
9.2 - Communication inter-processus 131
9.3 - Exemple 133
10 - Evolution des systèmes d'exploitation 134
10.1 - SE et travail lus ensemble 134
10.2 - SE résident 135
10.3 - SE avec zones de recouvrement 136
10.4 - Recouvrement des E/S lentes (SPOULE) 137
10.5 - Multiprogrammation 138
10.5.1 - SE multitâches (multitasking) 138
10.5.2 Système multi-utilisateurs 139
10.6 - Multitraitement 140
10.7 - SE multidimensionnel 142
10.7.1 - Traitement par lots à distance (RJE) 142
10.7.2 Temps partagé (time-sharing) 142
10.7.3 Transactionnel (transactional) 143
10.7.3.1 - Cas général 143
10.7.3.2 - Transactionnel et temps partagé 148
10.7.3.3 - Transactionnel distribué 148
10.7.4 - Télétraitement 149
10.7.5 - Temps réel (real time) 150
10.8 - Mémoire virtuelle 151
10.9 - Machines virtuelles 151
10.10 - Systèmes d'exploitation distribués 156
10.10.1 - Principes généraux 156
10.10.2 - Exemples 157
10.11 - Système d'exploitation orienté objet 160
11 - Gestion de la mémoire centrale 163
11.1 - Partitions fixes (statiques) 163
11.2 - Partitions de longueur variable 165
11.3 - Partitions translatables (dynamiques) 165
11.3.1 - Partitions avec recouvrements système 167
11.3.2 - Partitions avec code et données séparés 168
11.3.3 - Partitions multiples 169
11.3.4 - Remarques sur les recouvrements 169
11.4 - Pools centralisés 170
11.5 - Pagination 171
11.5.1 Support physique 171
11.5.2 - Cas des canaux 174
11.5.3 - Lien avec l'antémémoire 174
11.6 - Les types de registres de traduction 175
11.7 - Pagination statique 178
11.8 - Pagination dynamique 179
11.8.1 - Pré-pagination 183
11.8.2 - Post-pagination 184
11.8.3 - Choix de la page à vider 185
11.8.4 - Seuils, vidage et pagination 187
11.8.5 - Working set (WS) 190
11.8.6 - Pagination à deux niveaux ou plus 191
11.9 - Protection 193
11.10 - Chargement des programmes 193
11.11 - Exemples 193
11.11.1 - Pagination sous DOS/VS 193
11.11.2 - Pagination sous MVS 194
11.11.3 Pagination sous VAX/VMS 196
11.11.4 - Pagination sous VM 198
12 - Gestion de la mémoire virtuelle 200
12.1 - Espace d'adresses 201
12.2 - Segmentation 202
12.3 - Support matériel 204
12.4 - Gestion MC avec segmentation seule 205
12.5 - Protection 206
12.6 - Segmentation plus pagination 207
12.7 - Mémoire à un seul niveau 207
12.8 - Mémoire virtuelle distribuée 208

12.9 - Exemples de structure de MV 209
12.9.1 - Cas des DPS7 209
12.9.2 DOS/VS . 209
12.9.3 MVS 210
12.9.3.1 Adressage 24 bits (MVS/370) 211
12.9.3.1.1 Zone commune . 211
12.9.3.1.2 Zone système 212
12.9.3.1.3 - La zone privée 213
12.9.3.2 Adressage étendu 31 bits, MVS/XA . . 215
12.9.4 - VAX/VMS 216
12.9.5 - VM chez IBM 217
12.9.6 L'iAPX286 . . . 217
12.9.7 - GCOS8 218
13 - Fonctions réseau 220
13.1 - Réseaux en étoile 220
13.2 - Evolution hiérarchique 222
13.3 Cas des mini-ordinateurs 223
13.4 - Dialogues entre ordinateurs 223
13.5 - Architectures réseau 224
13.6 Services offerts . . . 227
13.7 - Modèle ISO 229
14 - Partage et protection 231
14.1 - Le problème posé 231
14.2 - Modèle général 231
14.3 - Protection de l'accès au système 232
14.4 - Modes de fonctionnement 233
14.5 Cas des multiprocesseurs . . . 234
14.6 - Protection en mémoire 236
14.6.1 Registres, clés et bits . 236
14.6.2 Pagination 237
14.6.3 - Pagination et anneaux 238
14.6.4 - Segmentation et anneaux 239
14.6.5 - Listes d'accès et capacités 244
14.6.6 - Utilisation des piles 246
14.7 - Sur mémoire périphérique 247
14.8 - Privilèges, limites, quotas 248
14.9 - Autres protections 248
14.10 - Partage 249
14.10.1 - Verrous et sémaphores 249
14.10.2 - Partage de données en mémoire . . . 250
15 Langages de commande 252
15.1 Types de langage . . . 252
15.2 Message d'invitation 252
15.3 Syntaxe des langages . 253
15.4 Procédures catalogées 255
15.5 - Règles de recherche 256
16 - Génération, installation, démarrage, configuration 257
16.1 - Démarrage du système 257
16.2 Installation 259
16.3 - Génération du SE 260
16.4 - Personnalisation du SE 261
16.5 - Exits 262
16.6 - Configuration 262
16.7 Arrêt du système 262
17 - Gestion des performances 263
17.1 - La démarche "performances" 263
17.2 - Les mesures en continu 264
17.3 Les mesures indispensables 266
17.4 - Mesures exceptionnelles 267
17.5 - Suivi des performances par les constructeurs 270
Annexe A - Systèmes d'exploitation sur IBM 370 273
Annexe B - Lexique 275
Annexe C - Liste des abréviations 276
Annexe D - Références 279
Annexe E - Index 290

introduction

Ce tome est le deuxième d'une collection consacrée à l'architecture des systèmes informatiques et organisée comme suit:

- tome I: l'architecture du **sous-système central**, décomposée en architecture externe ou fonctionnelle et architecture interne. Il couvre le processeur central et la mémoire centrale. Y sont présentés les jeux d'instructions, modes d'adressage, organisation des registres et des opérateurs, parallélisme et anticipation, systèmes à tolérance de panne, ainsi que les différentes architectures;
- tome III : **entrées-sorties**: description des processeurs d'entrée-sortie et canaux, des mécanismes d'entrée/sortie et d'interruption. On y trouvera aussi un chapitre détaillé sur les bus;
- tome IV : **périphériques**: y seront décrits les sous-systèmes périphériques, et tout particulièrement ceux qui assurent la fonction de stockage (bandes et disques);
- tome V : **développement de programme**, c'est-à-dire l'ensemble des transformations que subit un programme écrit dans un langage de haut niveau pour obtenir un programme en langage machine exécutable par le matériel de l'ordinateur. Les sujets abordés seront la compilation, l'assemblage, l'édition de liens, le chargement, la structure et l'utilisation des sous-programmes, les bibliothèques.

Ce tome II, principalement consacré au **système d'exploitation**, complète l'architecture du matériel du sous-système central en introduisant les éléments requis par le système d'exploitation (par exemple la pagination) lorsque leur présence devient nécessaire. D'autres compléments à l'architecture du sous-système central, comme les canaux, les interruptions ne seront présentés que dans le tome III, tome au bout duquel le lecteur aura une vue complète et structurée du sous-système central, aussi bien sur le plan du matériel que sur celui du logiciel.

L'élément central de ce tome II est le système d'exploitation, abrégé en SE (OS comme Operating System en américain). Les systèmes d'exploitation seront abordés sous un éclairage particulier dans la mesure où il existe un grand nombre d'ouvrages sur le sujet. Ces ouvrages sont en général:

- pour les uns **spécialisés**, consacrés à un seul système d'exploitation, dans le domaine des micro-ordinateurs en particulier,
- d'autres ont une orientation **universitaire**,
- peu font référence aux grands SE réellement utilisés dans le monde professionnel.

Ce tome II se veut complémentaire par rapport à ces ouvrages: il couvre les petits systèmes d'exploitation des micro-ordinateurs, mais aussi ceux des gros ordinateurs de gestion, il fait référence aux systèmes d'exploitation que l'on rencontre couramment dans l'industrie, la gestion, la recherche, et insiste peu sur les systèmes d'exploitation d'origine universitaire abondamment et brillamment décrits par ailleurs. Il constitue donc un pont entre le monde universitaire, le monde de l'écriture, et le monde professionnel et familiarise le lecteur avec le vocabulaire qu'il entendra dans le milieu des constructeurs et des utilisateurs. Cette complémentarité et sa limitation à environ 240 pages nous ont conduit à favoriser les exemples pratiques par rapport à l'exposé théorique.

Comme le tome précédent, ce tome est destiné à tous ceux qui désirent approcher de façon structurée les systèmes informatiques, afin de situer les objets dans un premier temps, d'en connaître les fonctions, les relations, puis d'en étudier le détail. Il intéresse:

- les enseignants et les étudiants,
- les professionnels de l'informatique voulant élargir ou approfondir leur domaine de compétence,
- les esprits curieux qui souhaitent comprendre les ordinateurs à l'aide d'une approche structurée et mieux les utiliser,
- tous ceux qui veulent suivre l'évolution des systèmes informatiques.

Il n'est pas destiné aux lecteurs totalement étrangers à l'informatique qui ne recherchent qu'une initiation, une sensibilisation indolores. Il se conçoit comme:

- support du cours donné par l'auteur, ou par d'autres enseignants,
- ouvrage de complément pour ceux qui ont suivi un premier enseignement, lu d'autres ouvrages d'initiation,
- aide à mieux comprendre leur environnement pour ceux, toujours plus nombreux, qui travaillent dans un environnement informatique hétérogène.

Ces différents tomes sont le fruit d'une longue expérience professionnelle et d'une réflexion poursuivie au contact de très nombreux publics: promotion supérieure du travail (CESI), Ponts et Chaussées, formation du personnel de l'Education Nationale, formation de base dans des écoles d'ingénieurs comme l'Institut d'Informatique d'Entreprise (IIE), entreprises et écoles supérieures de commerce (CERAM), écoles d'ingénierie (CICOM). Le contenu de ce tome a été enseigné par l'auteur aux différentes catégories citées ci-dessus, depuis une dizaine d'années. Il suit une démarche qui n'a cessé d'être remise en cause et améliorée au contact des différents publics cités et à la lumière de son expérience professionnelle.

Nous conserverons la méthode d'organisation, de classement utilisée dans le tome I et illustrerons les notions présentées par le sys-

tème d'exploitation le plus répandu lorsqu'il le permet, MS-DOS, popularisé par les PC d'IBM. Ce tome n'est cependant en aucun cas un ouvrage sur MS-DOS, SE par trop limité pour illustrer nombre de notions abordées. Il n'a rien d'un manuel utilisateur.

Dans le domaine des ordinateurs de gestion nous prendrons de préférence des exemples chez IBM, le constructeur dominant, ainsi que chez Bull, le constructeur national. Mais des machines aussi répandues que les VAX de DEC ne seront pas ignorées, ainsi que des systèmes comme Unix. Comme autre outil pédagogique nous utiliserons différents langages dont le **Cobol** qui est le plus répandu dans le domaine de la gestion, le **Basic** qui est connu du plus grand nombre.

Pour illustrer les notions matérielles présentées au cours des différents chapitres, nous utiliserons de préférence un des microprocesseurs appelés à se répandre dans le proche futur, l'**iAPX286** (ou 80286) d'Intel, que l'on commence à trouver sur de nombreux micro-ordinateurs, en particulier les compatibles du PC/AT d'IBM.

Ce tome est conçu pour être lu et **relu**, pour deux raisons principales:

- il n'est pas facile de donner des exemples sans faire référence à des notions expliquées par la suite, et tous les exemples n'intéressent pas tous les lecteurs avec la même intensité;
- certains chapitres sont beaucoup plus techniques et nous conseillons au lecteur qui recherche dans un premier temps une vue d'ensemble, la lecture des chapitres 1 à 4, ainsi que celle des chapitres 10 et 13.

Certains détails ne prendront toute leur valeur que lors de relectures vivement conseillées. Nous avons conservé le découpage avec de nombreux chapitres, l'introduction des notions est progressive.

Ce tome est écrit en **français** dans la mesure du possible, mais l'inflation des termes anglo-saxons rend la tâche plus difficile qu'elle ne le fut dans le tome I. Nous avons bien sûr évité de forger des termes nouveaux, mais nous nous sommes cependant permis d'étendre les termes officiels comme "spoule" par exemple pour en faire un verbe, un adjectif lorsque nécessaire. Du fait de l'erreur originelle, nous avons fréquemment traduit des termes comme "control" par "contrôler" alors qu'il signifie "commander", le purisme ne serait plus compris actuellement. L'importance des termes américains nous a conduit à les mettre entre parenthèses à la suite des termes français correspondants, au singulier ou à l'infinitif pour les verbes, et à adjoindre un court lexique d'une page en fin de tome.

Quelques schémas contiennent des mots américains, nous pensons qu'il aurait été maladroit de les traduire. Quelques mots bien français sont utilisés de façon particulière en informatique: ainsi "voit" dans l'expression "mémoire que voit le programmeur" (chapitre 11) a choqué un des membres du comité de lecture, membre pourtant expérimenté en informatique applicative.

Le problème des **abréviations** s'est aussi posé au cours des relectures. Certaines d'entre elles, comme SE par exemple, ne doivent pas provoquer de problème pour l'ensemble des lecteurs. D'autres comme WS ne gagneraient pas grand chose a être écrites en français, au

contraire. Les abréviations spécifiques d'un système d'exploitation ne gagnent pas non plus à être traduites en une expression française, elles seront comprises par les lecteurs qui s'intéressent plus particulièrement à ce système, les autres passeront dans un premier temps.

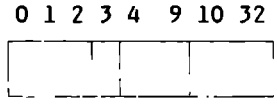
Les **références** bibliographiques figureront dans le texte sous forme de crochets contenant:

- les trois premières lettres du nom du premier auteur,
- les deux chiffres du millésime,
- une lettre éventuelle pour distinguer les doublons.

Les tableaux et **figures** sont repérés par le numéro du chapitre courant, suivi d'une ou de deux lettres (aa vient après z) minuscules.

Lorsque nous employons l'expression "OS d'IBM" pour "Operating System d'IBM", nous désignons des systèmes d'exploitation d'IBM bien précis, ceux tournant sur les séries 360, 370, 43X1, 303X, 308X et 3090, et plus particulièrement ceux appelés OS par IBM: MFT, MVT, VS1, VS2, MVS. Les systèmes d'exploitation DOS qui tournent sur ces mêmes ordinateurs sont exclus de l'expression "OS d'IBM".

Quand nous dessinons le format d'un **mot machine**, nous suivons le format du constructeur. Le numérotage des bits du mot, généralement donné au-dessus du mot se fera donc en partant de zéro ou de un, en commençant par la gauche ou par la droite. Le lecteur ne doit pas s'en alarmer, il ne s'agit que d'une convention qui ne joue aucun rôle dans ce tome:



exemple d'un mot de 32 bits, numérotés de gauche à droite en partant de zéro, avec quatre champs dans le mot, le premier de 3 bits, le second d'un d'un seul bit, le troisième de 6 bits, etc.

oOo

Nous tenons à remercier particulièrement M. *Pierre Marquès* pour les conseils et l'aide essentiels qu'il a continués à nous apporter tout au long de la réalisation de ce tome, son expérience, acquise au cours d'une longue carrière dans l'imprimerie nous faisant totalement défaut. Notre reconnaissance s'adresse aussi au lieutenant-colonel *Denis Fargette*, MM. *Jean-Yves Collot*, *Jean-Pierre Djon*, *Pierre Montagnon*, *Alain Pinet*, *Raymond Poedras*, qui ont bien voulu relire tout ou partie de ce tome et nous faire bénéficier de leurs remarques. Nos remerciements vont encore à M. *Jean-Marie Dubois* qui a consacré de nombreuses heures à revoir toutes les références à MVS; nous espérons que notre exposé ne trahit pas, par manque de place, la précision et la clarté de ses explications. M. le *Professeur Claude Kaiser* mérite aussi toute notre gratitude, non seulement pour avoir bien voulu rédiger la préface, mais pour ses conseils et recommandations; les contraintes de nombre de pages n'ont malheureusement pas permis de les prendre tous en compte.

chapitre 1

définitions et généralités

1.1 – Définition générale.

La définition la plus générale que l'on puisse donner d'un système d'exploitation est celle qui le présente comme une extension logicielle du matériel, dans le but d'offrir un niveau de service suffisant aux utilisateurs. Cette définition a l'avantage de:
- faire apparaître le **matériel**, ressource sur laquelle s'appuie le système d'exploitation (SE);
- citer les **utilisateurs** qui constituent la finalité du SE;
- préciser que le système d'exploitation fait partie du logiciel.

Elle dépend cependant de notions relativement floues comme:
- le "niveau de service suffisant";
- les utilisateurs dont les caractéristiques ne sont pas précisées.

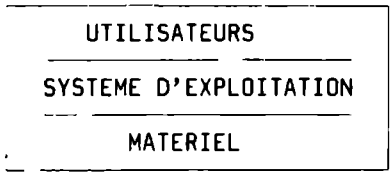


Figure 1a: situations respectives du matériel, du SE, des utilisateurs. Matériel plus SE sont parfois traités de "machine étendue".

1.2 – Les utilisateurs du SE.

La notion d'utilisateur est liée à l'endroit où on se place et on emploie couramment le terme d'utilisateur final pour parler du "vrai" utilisateur, de celui qui n'est pas un informaticien. Nous distinguerons trois catégories d'utilisateurs:
- ceux du système d'exploitation,
- les utilisateurs finals "directs", ceux qui ont un contact direct avec l'ordinateur, au moyen de postes de travail du type écran-clavier. Ils utilisent par exemple des machines de traitement de texte (TdT), des outils dits "infocentre", des applications transactionnelles, etc.,
- les utilisateurs finals "indirects" qui n'ont pas de contact direct avec l'ordinateur, mais en utilisent les sorties comme les états

imprimés par exemple, les graphiques, etc. Ils ne sont pas amenés à manipuler l'ordinateur.

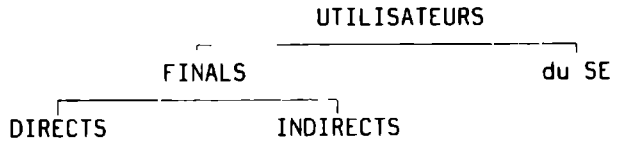


Figure 1b: types d'utilisateurs.

L'utilisateur du SE se présente comme un utilisateur intermédiaire puisque la finalité du système informatique est d'offrir des services aux utilisateurs finals. Le SE offre donc des services à des utilisateurs spécialisés dont la fonction principale est de:
- réaliser des applications pour les utilisateurs finals,
- gérer le système informatique lui-même.

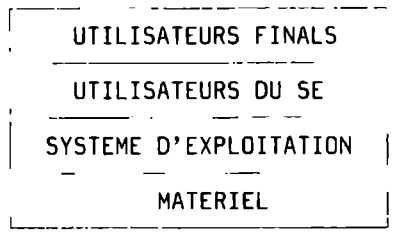


Figure 1c: utilisateurs finals et utilisateurs du SE.

Comme pour tout service, nous voyons apparaître:
- le service lui-même, celui offert par le "système informatique", c'est-à-dire l'ensemble des moyens informatiques,
- ceux qui le gèrent, à savoir une partie de ceux que nous avons appelés utilisateurs du SE,
- les utilisateurs du service.

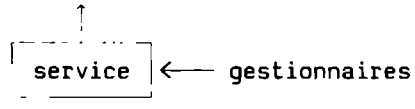


Figure 1d: le SE en tant que service.

1.3 – Structure générale du logiciel.

Cette structuration des utilisateurs a une incidence directe sur celle du logiciel. Elle conduit à décomposer le logiciel en:
- logiciel de **base**,
- logiciel d'appplication, qualifié aussi d'**applicatif**.

L'idée qui correspond au terme de "logiciel de base" est celle d'un ensemble de logiciels nécessaires pour fournir le niveau de service que nous avons évoqué. Parmi ces logiciels certains ont toutes les caractéristiques des logiciels applicatifs:
- ils s'exécutent de la même façon, sont chargés par le chargeur du SE,
- ils passent par les phases habituelles de production de programme: compilation, édition des liens.

Une partie de ces logiciels correspond à ce que l'on appelle des **utilitaires.**

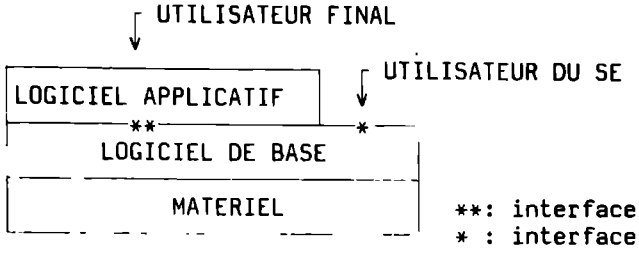
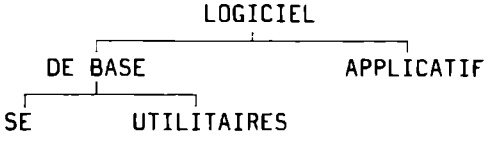


Figure 1e: première structuration du logiciel.

Dans ce contexte, le système d'exploitation est ce qui reste du logiciel de base lorsque l'on en extrait ces utilitaires. Cette structure correspond à la définition que l'on trouve dans un cours RSX11M de DEC: "Le SE se décompose en deux parties, l'exécutif et les utilitaires.", le terme de SE étant utilisé à la place de logiciel de base et exécutif à la place de SE.



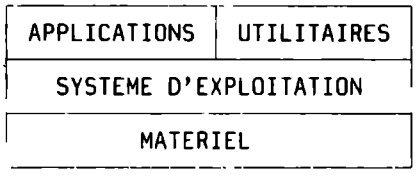


Figure 1f: première structuration du logiciel de base.

Un autre cours sur les SE les définit en précisant qu'ils "ne comprennent pas les programmes des utilisateurs (les logiciels applicatifs), les compilateurs, les programmes de service tels que tri, copie, etc.". [DEN84] définit le SE par le fait qu'il "fournit aux autres programmes les éléments dont ils ont besoin pour fonctionner".

[MUL83A] définit un SE comme "un ensemble de programmes qui résident dans un ordinateur. Il interprète les instructions de l'utilisateur, pilote le matériel, (...) Il est aussi responsable du partage des ressources entre plusieurs utilisateurs". [PHI86] définit le logiciel système de DRM comme un "SE contrôlant l'exécution des applications et fournissant le contrôle des ressources matérielles, plus un ensemble d'utilitaires et de bibliothèques d'exécutables (run time support)". Le cours RTE-A de HP décrit un SE comme:

- une interface destinée aux utilisateurs qui les isole des détails de bas niveau concernant le matériel et les entrées/sorties,
- assurant le contrôle des ressources,
- gérant les interruptions,
- offrant un système de gestion de fichiers.

On trouve encore dans [IBM80C]: "Le but principal d'un SE est de fournir des transitions de travail à travail: le SE lit un travail,

l'exécute jusqu'à son terme, lit un autre travail, l'exécute, etc.". Nous développerons cette vue dans la présentation de la gestion des travaux. Nous n'utiliserons pas dans ce tome l'expression "système opératoire", synonyme de SE, peu employé dans le langage informatique courant.

1.4 – Les versions d'un logiciel.

Tout logiciel parcourt ce que l'on appelle un **cycle de vie** (life cycle) qui regroupe tous les événements le concernant. On classe les différentes étapes du cycle de vie de la façon suivante :

1. définition des besoins,
2. spécifications fonctionnelles,
3. analyse détaillée,
4. implémentation (codage),
5. intégration,
6. test,
7. documentation,
8. production (deployment),
9. maintenance.

Au cours de sa vie le produit évolue et il est indispensable d'identifier précisément chacun de ses états stables. Ces étapes marquantes sont appelées **versions** (release, version), elles correspondent généralement à l'ajout de nouvelles fonctions et sont repérées par un numéro, une lettre, un chiffre, un nom plus généralement comme SR2500 (Software Release 2500 de GCOS8). Une version dure plusieurs mois, couramment 6 mois, un an ou plus. Une même version fait l'objet de **mises à jour** (révision, update) pour corriger les erreurs, ajouter quelques améliorations; ces mises à jour sont repérées elles aussi par un numéro (ou une lettre), séparé du numéro de version par un point. On parlera ainsi de CP/M 2.2, CICS 1.6 par exemple.

Il existe des logiciels avec trois numéros, chez IBM en particulier, lorsque le premier numéro a été ajouté par la suite pour différencier de grandes évolutions de logiciel: ainsi il y a eu MVS/SP, puis il a fallu différencier SP1 de SP2; ces numéros différencient maintenant deux produits appelés aussi MVS/370 et MVS/XA, avec chacun leurs numéros de version et de mise à jour à deux chiffres. Dans le cas d'IMS on aura IMS 1.1.3, les deux derniers numéros jouant les rôles de révision majeure et mineure. Le troisième numéro est aussi appelé "niveau de modification" (modification level dans VSE). Les versions successives offrent généralement une **compatibilité** ascendante: une application s'exécutant sous la version n-1 du SE tournera sous la version n. Cette compatibilité ascendante n'est cependant pas toujours assurée.

La forme "source" du SE ou "le source du SE", expression utilisée pour parler du SE tel qu'il est écrit en langage de programmation par opposition à ses formes binaires, est de moins en moins fréquemment fourni aux utilisateurs. Dans le cas de DOS d'IBM, il n'est plus fourni depuis VSE/SP2.1.0.

1.5 – Portabilité, pérennité.

Un logiciel fourni par un constructeur peut avoir plusieurs états. Le logiciel **standard** est un produit livré sous la **même forme**, avec les mêmes fonctionnalités à tous les utilisateurs qui veulent l'acquérir. Il dispose éventuellement d'**options** que l'utilisateur acquiert ou non, mais ce choix ne modifie pas le contenu du produit de base. L'engagement du fournisseur à **supporter** un logiciel standard, c'est-à-dire à en suivre son utilisation, corriger les erreurs, l'améliorer, n'est cependant pas éternel et un constructeur comme IBM sera amené à préciser à ses clients quels sont ses produits stratégiques, c'est-à-dire ceux qu'il compte supporter "longtemps". C'est ainsi qu'à l'heure actuelle DOS/VS semble perdre de l'importance pour IBM, alors que MVS et VM sont des produits stratégiques.

1.6 – Facturation.

Jusqu'en 1978/1979 le logiciel était facturé avec le matériel par les constructeurs. Pour lutter contre les fabricants de matériels compatibles qui prenaient une part croissante du marché, IBM a alors séparé la facturation du matériel et du logiciel (unbundling). La grande majorité des autres constructeurs a peu à peu suivi la même politique.

Tableau 1g (en $)	L/PC	COUT INIT	LOCATION MENSUEL.	MAINTE-NANCE	COUT TOTAL SUR 3 ANS FEV 80	FEV 85	JUIN 85
MVS/SP-JES2, version 2	1	12840	4280	673	37634	151368	161962
MVS/XA Data Facility P.	1	1590	530	164	5058	21025	22485
DFSORT release 7	1	0	247	19	5449	7497	8009
Assembler H, version 2	2	813	271	14	nc	8800	9394
VS Cobol avec librairie	2	11232	1872	106	14211	66199	70785
VS Fortran avec lib.	2	1305	435	34	10237	14500	15425
RMF version 3.3	1	2250	750	67	7527	25072	26848
RACF	1	0	786	41	15054	24899	24899
ACF/NCP version 3	1	12550	425	148	3613	18508	19802
IMS/VS version 2	1	0	8900	1465	77316	312064	312064
VM/XA	1	11220	3740	623	32064	133429	142579
TOTAL sans VM		32580	18496	2731	176099	649931	671672
TOTAL avec VM		43813	22238	3357	208163	783360	814251

Un logiciel peut s'acheter (acquisition d'une licence) ou se louer selon un coût mensuel lié à la période pendant laquelle on s'engage à utiliser le produit. Ceci correspond au droit (license) d'utilisation. Il peut y avoir de surcroît un coût mensuel de maintenance qui permet à l'utilisateur de bénéficier de l'assistance du constructeur en cas d'ennui et de profiter de l'évolution des logiciels. Ces coûts sont fréquemment dégressifs à partir de la seconde copie acquise par un même client, la seconde copie dans un même site (le terme de licence par processeur est quelque peu ambigu:

il correspond en fait aux multisystèmes dans lesquels il y a plusieurs copies du SE). En octobre 1986, IBM a introduit une facturation proportionnelle à la puissance du matériel avec la série 9370, afin de ne pas handicaper les petites configurations.

Dans le tableau 1g [DJU85] montre, d'après des chiffres de Annex Research, l'évolution du coût des logiciels chez IBM. Entre 1980 et 1985 ces coûts ont été multipliés par quatre environ; L/PC y correspond à une licence par processeur.

Les revenus des constructeurs ont évolué en fonction de la facturation séparée et le logiciel représente 4,9 % des revenus d'IBM en 1982, 7 % en 1984 selon Annex Research. Ce taux passerait à 32 % en 1995 (63 milliards de dollars) selon [LMI86], il est de 8,4 % selon [ORD86] qui mentionne que le rapport des recettes sur les investissements en logiciel a été de 3,9 en 1983, de 4 en 1984, et de 5,4 en 1985 pour IBM. A titre d'exemple, Datamation de mai 1973 indiquait que le développement de l'OS/360 avait coûté 200 millions de dollars. Selon [WEI81] un constructeur dépensait en 1975 de 2 à 10 milliards de dollars par an pour programmer et développer un SE.

Tous les logiciels utilisés ne le sont pas toujours en toute légalité. [CW85A] avance le taux de 50 % de copies non autorisées dans le domaine de la bureautique (office software).

1.7 – Les fournisseurs.

Historiquement les fournisseurs de logiciel étaient les constructeurs d'ordinateurs: selon [BUR65], le marché du logiciel était évalué à 1,5 milliards de dollars en 1964, les trois quarts couverts par les constructeurs; dans le quart restant, 10 millions de dollars représentaient le service bureau IBM. Les SSII ont depuis pris une bonne part du marché, d'abord dans les progiciels, puis dans les sous-systèmes, transactionnels ou interactifs par exemple, dans les gestionnaires de bases de données, dans les systèmes d'exploitation eux-mêmes. Jusqu'aux fabricants de semiconducteurs qui font du logiciel: un tiers du chiffre d'affaires de Intel en 1986 est représenté par des cartes Multibus et du logiciel.

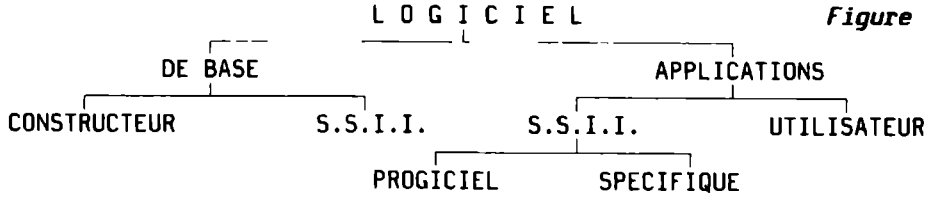


Figure 1h.

1.8 – Importance du logiciel.

L'évolution des systèmes d'exploitation s'inscrit dans celle plus générale des logiciels à laquelle on applique couramment le retournement de la proportion des 80/20: alors que par le passé les coûts relatifs se répartissaient grossièrement en 80 % de coûts matériels

contre 20 % pour le logiciel, le logiciel représente maintenant 80 % là où le matériel compte pour 20 %.

Le gouvernement américain dépensait, en 1981, 6 milliards de dollars par an pour ses dépenses de logiciel, et le GAO (organisme gouvernemental américain) estimait alors qu'en 1985 le logiciel représenterait environ 90 % du budget informatique fédéral ([HYS81]). Une estimation IDC de 1981 prévoyait pour 1985 une dépense de 8,4 milliards de dollars de logiciel sur les gros/moyens ordinateurs (mainframes).

Un autre élément soulignant l'importance du logiciel est lié à la **pénurie de programmeurs** (programmer shortage) qui réapparaît de temps à autre. Ainsi le Japon estime manquer de 600 000 programmeurs.

Une enquête conduite auprès de 642 responsables informatiques américains donne dans [VER85] les résultats suivants pour le budget informatique:

en % selon [VER85]

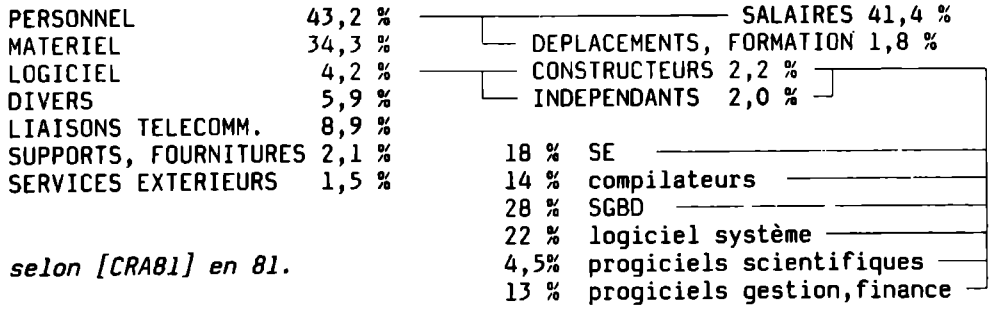
		LOGICIEL		
	PERSONNEL	APPLICATIF	SYSTEME	MICRO
1984	37,8	2,6	1,3	4,3
1985	38,0	2,5	1,5	4,5

Ces chiffres doivent être comparés à la prévision d'un des gourous de l'informatique: James Martin prévoyait, à la fin des années 70, l'évolution ci-dessous:

en %

	SALAIRES	MATERIEL	LOGICIEL	TRANSMISSION	DIVERS
1977	30	40	10	9	4
1982	50	10	21	15	4

[CRA81] avance 45 à 50 % dans les années 72 à 80 pour les coûts des personnels informatiques dans le budget informatique total. La part des logiciels et des télécommunications augmentaient de 20 % par an pendant cette période, les dépenses informatiques correspondant à 1,33 % des revenus de l'entreprise (corporate):



selon [CRA81] en 81.

Les chiffres ci-dessus correspondent généralement à de grandes entreprises. Selon [LS86B] l'ADI estime le marché du logiciel en France à:

VENTES 1984 EN MILLIARDS		CROISSANCE ANNUELLE 1982 1984 EN %
MATERIELS	43	19
PROGICIELS		
CONSTRUCTEURS	3,1	30 à 40
INDEPENDANTS	2,2	30 à 40
LOGICIELS,SERVICES,CONSEIL	30,7	28

[THO85] avance le chiffre de 37,3 milliards de francs dépensés en France en 1983 pour le logiciel, contre 30,3 pour le matériel.

1.9 – Langages d'écriture des systèmes d'exploitation.

Initialement les systèmes d'exploitation étaient écrits en langage d'assemblage, et ceci a duré beaucoup plus longtemps que pour les applications car les programmeurs système étaient coutumiers de ce type de langage et parce que la performance maximum était recherchée au niveau des SE. L'environnement restait plus technique. Les problèmes de maintenance, de productivité des programmeurs, ont poussé les constructeurs à utiliser de plus en plus des langages de haut niveau pour développer les systèmes d'exploitation. Ils conservent quelques séquences en langage d'assemblage lorsque les instructions des langages de haut niveau ne permettent pas d'activer le matériel ou lorsqu'il y a des problèmes de performance. Le tableau ci-dessous en donne quelques exemples, Burroughs étant un pionnier dans ce domaine:

BURROUGHS	ESPOL sur le B5000 en 1962	PRIME	FORTRAN puis PL/1(G)
MULTICS	PL/I		
UNIX	C à 90 %	GCOS7	MLP et PL/1
ICL 2900		MVS	PLS
CP VI	PL-6 dialecte PL/I	NORSK DATA	NORD-PL

1.10 – Conclusion.

Une définition précise et universellement admise pour un SE n'existe pas (cf. [PET83] page 3). Dans [CII76] on trouvera "qu'il est parfois difficile d'obtenir un accord général sur ce qui fait ou ne fait pas partie du SE". De même [AND81] note que "bien que tous les ordinateurs aient une forme de SE, dire précisément que tel module logiciel en fait partie est une affaire d'opinion.".

Le SE sera donc pour nous une couche de logiciel qui se situe entre le matériel et les logiciels applicatifs, offrant à ces derniers et aux utilisateurs du SE un service jugé acceptable. Le SE joue donc un rôle intermédiaire entre le matériel et ses utilisateurs (individus ou programmes), rôle qui se traduit par ses fonctions de "gestionnaire de ressources" que nous détaillerons plus loin, rôle que l'on retrouve dans son nom: système assurant l'exploitation des ressources.

chapitre 2

fonctions du SE

2.1 – Les fonctions générales.

Afin d'offrir ses services, le SE dispose des fonctions suivantes:
- la gestion des **requêtes** des utilisateurs;
- la gestion des **ressources** sur lesquelles il s'appuie pour satisfaire les requêtes des utilisateurs;
- la **gestion** et le **contrôle** du système informatique lui-même, c'est-à-dire les fonctions de pupitrage, de génération, de suivi comptable, d'optimisation, etc.

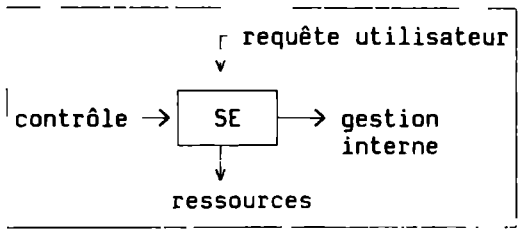


Figure 2a: environnement du système d'exploitation.

Selon les SE, ces fonctions seront offertes avec des caractéristiques de:
- disponibilité, sécurité, reprise sur erreur, plus ou moins grandes;
- facilité d'accès, richesse en fonctionnalités, plus ou moins développées.

On trouve ainsi dans [CII76]: "Le logiciel de base est conçu pour utiliser de façon efficace et commode le matériel. Il augmente les possibilités du système considéré comme un tout. Ses objectifs sont de:
- éviter à l'utilisateur d'avoir à considérer les caractéristiques détaillées du système;
- aider l'opérateur et le responsable du site;
- maximiser le débit pour utiliser le plus efficacement possible toutes les ressources du système.".

2.2 – La gestion des ressources.

La gestion des ressources suppose une modélisation du système informatique qui fait apparaître des:

- **acteurs**, sujets, agents ou objets actifs: ce sont les objets qui représentent les requêtes des utilisateurs et que l'on appelle généralement des **processus** (process), tâche (task), travail (job), etc.; ce sont des "consommateurs" de ressources;
- objets **passifs** qui sont **attribués** (alloués) aux acteurs, selon leurs besoins, par le SE. Parmi ces objets appelés **ressources** on trouvera l'espace sur disque, l'espace en mémoire, le temps processeur, etc.;
- un élément **décisionnaire** qui décide de l'attribution de ces ressources aux acteurs: le SE qui se base sur des critères qui lui sont propres et/ou définis par les utilisateurs (du SE).

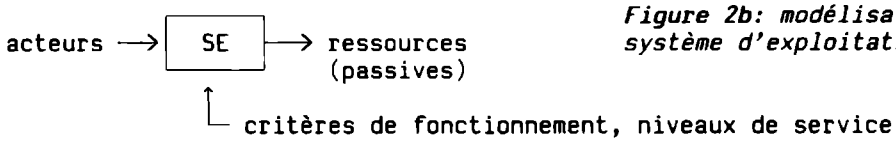


Figure 2b: modélisation du système d'exploitation.

2.2.1 – Classification des ressources.

On distingue généralement les ressources physiques des ressources logiques. Ces dernières ne peuvent exister sans supports physiques: un fichier n'a pas d'existence réelle sans un volume sur lequel il est stocké, un processus n'a pas de réalité en dehors d'un processeur.

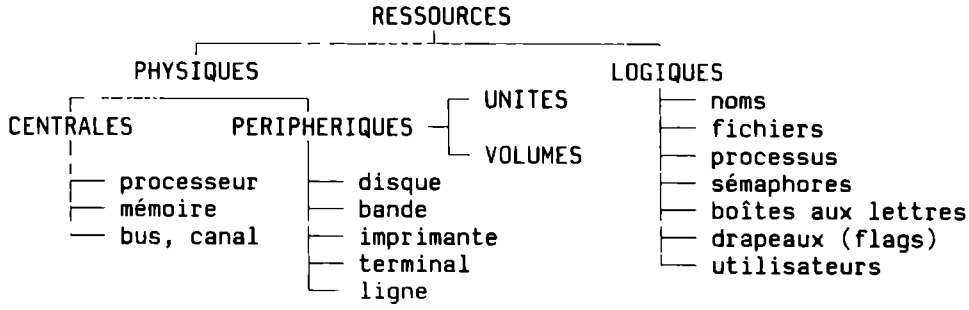


Figure 2c: principaux types de ressource.

2.2.2 – Mécanismes d'attribution des ressources.

L'attribution d'une ressource à un acteur se déroule en deux étapes successives:

- le SE analyse d'abord si l'utilisateur a le droit d'accéder à la ressource en se basant sur des informations fournies au moment de la

requête et/ou déduites d'une action précédente comme l'identification de l'utilisateur (procédure de connexion ou "login"),
- si l'utilisateur initiateur de la requête a le droit d'accéder à la ressource, le SE décidera ensuite à quel moment (QUAND), et pendant COMBIEN de temps la ressource sera attribuée à cette requête.

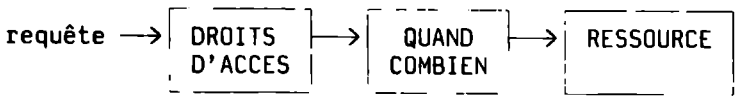


Figure 2d: mécanisme d'attribution des ressources.

2.3 – Gestion du système informatique.

2.3.1 – Organisation d'un service informatique.

Pour présenter les fonctions de gestion du système informatique, il convient de préciser l'environnement humain et organisationnel du service informatique qui gère la ressource globale que constitue pour l'entreprise l'ordinateur, ses logiciels, le personnel informatique.

Nous décomposerons le service informatique en un certain nombre de cellules dont quelques-unes peuvent être confondues pour différentes raisons dans certains contextes réels.

La cellule d'**exploitation**, parfois organisée en deux ou trois équipes lorsqu'il faut assurer un service sur plus de huit heures par jour, assure des tâches opérationnelles:
- le **pupitrage** à l'aide d'un poste de travail, un écran/clavier habituellement, et d'un langage particulier génériquement appelé OCL (Operator Command Language): le pupitreur peut interroger le système d'exploitation pour connaître l'état des ressources, pour lister les acteurs, les requêtes. Il peut aussi modifier l'état de ces objets, donner des ordres au SE;
- le **montage**/démontage des volumes comme les bandes magnétiques, les disques amovibles, l'alimentation des appareils du type lecteur de cartes ou imprimante, le déliassage, etc. Les **opérateurs** sont chargés de ces opérations;
- la **préparation** des travaux avant leur soumission. Elle consiste à rassembler en temps voulu tous les supports nécessaires à la bonne exécution de la requête de l'utilisateur;
- la **sauvegarde** des logiciels et données selon les consignes.

La cellule **système** (support technique, assistance technique) est responsable du maintien du système en bon état de fonctionnement. Elle joue les rôles suivants:
- un rôle quotidien de **maintenance** du SE qui consiste à corriger ou contourner les incidents qui se produisent et que la cellule opérationnelle n'a pu résoudre;
- un rôle de **suivi** des problèmes connus et dont la solution n'a pu être trouvée par l'équipe système qui a du les confier au fournisseur du SE;
- la préparation et la mise en place des nouvelles **générations** du SE;

- un rôle d'observation et d'**optimisation** afin que non seulement le SE fonctionne correctement, mais qu'en plus il fonctionne au mieux de ses possibilités;
- **l'étude et le test de nouveaux produits;**
- un rôle de développement de programmes de service, ou utilitaires pour faciliter l'utilisation et la gestion de l'outil informatique;
- l'écriture de séquences de commandes du SE, fréquemment appelées **procédures** cataloguées afin de faciliter le travail des utilisateurs qui ne sont pas spécialisés dans ce langage de commande (JCL comme Job Control Language).

Les membres de ce service sont appelés "hommes système" (programmeur système aussi), le responsable portant parfois le nom d'administrateur du système (system administrator dans [MUL83A]).

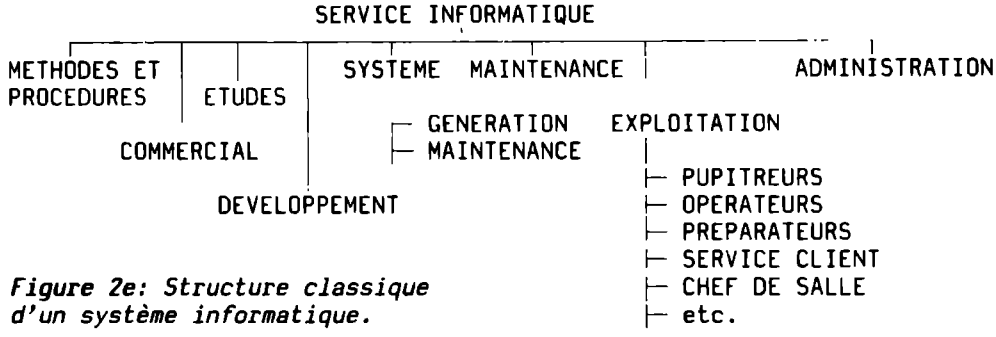


Figure 2e: Structure classique d'un système informatique.

La cellule **développement** réalise les applications, les programmes propres à l'entreprise.

La cellule de **maintenance** est chargée d'assurer le bon fonctionnement des applications de "production", c'est-à-dire des logiciels applicatifs qui ont été déclarés opérationnels.

La cellule **études/planification** diffère de la cellule système dans la mesure où elle est plus éloignée des outils informatiques, elle ne requiert pas nécessairement des techniciens spécialisés dans ces outils. Elle a pour rôle d'analyser le long terme. Dans certains environnements cet aspect stratégique est gommé et les études sont confondues avec le système et/ou le développement.

La cellule **administrative** s'occupe des achats et location de matériels ou logiciels, de l'inscription des utilisateurs, de la gestion des droits, de la documentation, de la **facturation** des services informatiques que ces derniers utilisent, de la gestion du parc.

La cellule **méthodes et procédures** est chargée de définir les règles selon lesquelles fonctionne le système informatique, sur les plans techniques et administratifs.

On voit apparaître aussi une fonction **commerciale** chargée de vendre en interne, dans l'entreprise, les services offerts par l'informatique et d'en faire la promotion.

Du type de SE peut dépendre la taille de l'équipe informatique capable de le "supporter": [AND86] cite l'exemple d'une société améri-

caine qui passa d'une équipe de 20 personnes à un groupe de 8 personnes en 1984 en passant d'un 4341-2 à un System/38; l'équipe de programmation passa de 10 à 3 personnes, plus productives que les 10 réunies auparavant.

2.3.2 – Fonctions de gestion du système.

Le SE, ou plus précisément le logiciel de base dans le cadre de ce tome offre généralement les fonctions suivantes pour assurer une bonne gestion du système informatique:

- des outils facilitant l'**installation** du SE;
- des outils de **mesure** et de **suivi** (logimètres logiciels) facilitant l'optimisation, le bon réglage du système;
- des outils de **test** comme CHKDSK sur MS-DOS;
- des **utilitaires** de **sauvegarde**/restauration (save/restore) des fichiers, ceux contenant le SE compris:
  * **logique**: chaque fichier est sauvegardé indépendamment. L'équivalent sur MS-DOS est la commande COPY;
  * **physique**: la décomposition en fichier n'est pas respectée, chaque unité physique (bloc, secteur, piste) est copiée en séquence, quelque soit le fichier auquelle elle appartient. DISKCOPY sur MS-DOS correspond à une telle copie physique, les commandes BACKUP et RESTORE étant réellement les commandes de sauvegarde-restauration du disque dur;
  * **incrémentale**: tout n'est pas sauvegardé, seulement ce qui a été modifié. On distingue les sauvegardes incrémentales explicites dans lesquelles seuls les fichiers modifiés sont sauvegardés (un bit est positionné à un lors de la sauvegarde par exemple, puis remis à zéro dès que le fichier est modifié, cas de la commande BACKUP de MS-DOS avec le bit ARCHIVE), des sauvegardes incrémentales périodiques automatiques où le système vient sauvegarder ce qui a été modifié au cours du dernier intervalle (cas de Multics);
  * **partielle**: il est possible de ne sauvegarder qu'une partie des fichiers, une sous-arborescence par exemple.
- des informations **comptables** que le SE stocke régulièrement sur le fichier comptable;
- des outils de **maintenance** comme PCtools sur MS-DOS;
- la fonction de **pupitrage**.

chapitre 3

les types de SE

3.1 – Typologie.

Les requêtes acceptées par les systèmes d'exploitation ont des formes différentes selon leur origine et le mode d'utilisation:

- celles provenant des utilisateurs humains seront:
  * soit formulées au coup par coup sous forme de **commandes**, l'utilisateur pouvant tenir compte des résultats de la commande en cours avant d'entrer (de "frapper") la commande suivante. Ce mode de fonctionnement est appelé **mode dialogué**, ou moins officiellement "interactif", "conversationnel";
  * soit regroupées en ensembles de commandes appelés **travaux**. Ces travaux sont stockés par le SE puis exécutés ultérieurement selon des critères de **planification** que nous examinerons. On parlera alors de **mode différé** ou de traitement par **lots**, un lot étant un ensemble de travaux provenant de différents utilisateurs.
- celles provenant d'appareils, appareils de mesure ou autres capteurs, machines-outils que pilote l'ordinateur. C'est le domaine du **temps réel**;
- les requêtes peuvent elles-mêmes créer de nouvelles requêtes durant leur exécution. Un travail peut en soumettre un autre, une transaction peut en lancer une autre, etc.

Citons, dans un but exhaustif, les requêtes que peut soumettre le pupitreur, autres que les commandes qui lui sont propres (commandes OCL), et qui se présentent pour le SE sous la même forme que les requêtes des utilisateurs.

Bien que visant l'universalité, les SE ont des caractéristiques qui les rendent plus adaptés à certains environnements. On peut bien entendu travailler en mode dialogué sous GCOS3, mais de façon moins agréable que sous Multics. De même dans l'environnement IBM, CMS sous VM est beaucoup plus apprécié que TSO sous MVS ou ICCF sous DOS, encore que pour certains cela relève de guerre de chapelles: CMS est plus utilisé en environnement scientifique pour développer et tester des fonctions système, chacun étant isolé des autres utilisateurs; TSO

est plus adapté à l'environnement de gestion, avec de nombreux fichiers accessibles répertoriés dans une structure arborescente [DUB86]. Dans le monde des micros iRMX86 d'Intel permet de faire du développement en interactif, mais le fonctionnement multi-utilisateur laisse alors à désirer.

De par leur conception de base nous distinguerons trois grands types de systèmes d'exploitation:

- ceux orientés vers le mode **différé**, le traitement par lots (batch processing en américain), comme MVS d'IBM, GCOS3 de Bull;
- ceux visant le mode **dialogué** ou traitement interactif, comme Multics de Bull, Unix d'AT&T, VMS de DEC, CMS d'IBM;
- ceux adaptés au **temps réel** (real time) comme iRMX86 d'Intel, RT-11 de DEC, RTE-A de HP.

3.2 – SE orientés vers le mode différé.

Ces SE sont généralement conçus pour des environnements de **production**, environnements opposés à ceux de développement de programme, d'éducation, de recherche, etc. Ils correspondent sur le plan du matériel aux gros et moyens ordinateurs de gestion.

Leur objectif principal est de traiter un maximum de requêtes en optimisant **l'utilisation des ressources**, c'est-à-dire en cherchant à les occuper le plus possible, en visant des taux de 100 %.

Cet objectif peut être atteint dans la mesure où la **charge** (l'ensemble des requêtes des utilisateurs, workload) que ces SE doivent traiter est **contrôlable**: le SE décide des instants où il va accepter de prendre en compte telle ou telle requête préalablement stockée. Il pourra ainsi charger la machine avec des requêtes, des travaux complémentaires, l'un étant apte à saturer le processeur (programme de calculs scientifiques par exemple), l'autre la mémoire, un troisième les disques, etc. L'ensemble des ressources est ainsi activé simultanément et le rendement global est optimum.

La forme sous laquelle un utilisateur soumet une requête est celle d'un **travail** (job), c'est-à-dire d'un ensemble de commandes comprises par le SE, et de données comprises par les applications ou utilitaires activés par ces commandes. Les commandes ainsi utilisées font partie du langage de commande des travaux (JCL comme Job Control Language généralement, JCS comme Job Control Statement dans DOS/VSE).

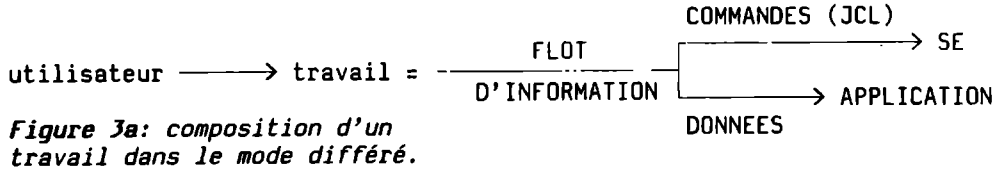


Figure 3a: composition d'un travail dans le mode différé.

Le travail est un ensemble d'**étapes** (step chez IBM, activity sous GCOS3) déroulées séquentiellement, l'une après l'autre, dans l'ordre où elles sont soumises. Chaque étape correspond à l'exécution d'un programme sous sa forme exécutable (cf. tome V).

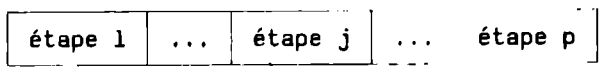


Figure 3b: structure d'un travail.

Chaque étape est matérialisée dans le flot d'information par une commande particulière: // EXEC sous MVS, $ EXECUTE, $ CBL74, $ PROGRAM sous GCOS3, $ STEP sous GCOS7.

Les services du SE ne sont généralement pas sollicités séparément pour chaque travail, mais pour un ensemble de travaux regroupés dans ce que l'on appelle un **lot** (batch), d'où l'expression consacrée de traitement par lots attribuée à ce mode de fonctionnement.

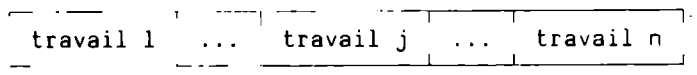


Figure 3c: structure d'un lot.

Le travail est a priori une notion purement **administrative** regroupant tout ce qui concerne un même utilisateur ou un même centre de coût. On saura ainsi vérifier ce à quoi il a droit, les résultats de ses différentes étapes seront plus facilement regroupés, etc. S'il correspond à une notion comptable, toutes les ressources consommées par le travail seront facturées à l'utilisateur concerné. Le travail porte généralement un nom: "//nom-travail JOB" sous les OS d'IBM (les deux barres obliquent indiquent qu'il s'agit d'une commande destinée au SE et non pas de données destinées à une application). Dans GCOS3 par contre, en dehors du traitement par lots, le travail n'a pas un nom choisi par l'utilisateur, le même à chaque exécution, mais un numéro ou une chaîne de caractères attribué selon un algorithme propre à une installation donnée ou bien fixé par la commande "$ SNUMB nom"; l'affectation comptable peut y être modifiée dans le travail par la commande $ IDENT et la consommation ainsi répartie sur plusieurs numéros de compte. De même le paramètre ACCT de MVS qui peut se mettre au niveau de chaque étape.

Le travail n'est cependant pas totalement limité à ces notions d'identification et de comptabilité puisque dans de nombreux SE les ressources peuvent être conservées entre étapes d'un même travail (cf. chapitre 5).

Pour l'utilisateur la **qualité du service** fournie par ce type de SE se traduit, non par le taux d'utilisation des ressources, mais par le temps de réponse, c'est-à-dire le temps qui s'écoule entre l'envoi de la requête et le retour des résultats (turnaround time).

Initialement l'utilisateur déposait au guichet de la salle machine son travail sous la forme d'un paquet de cartes. Ce travail était ensuite regroupé avec d'autres travaux pour former un lot que l'opérateur plaçait dans le lecteur de cartes. Le SE lisait ensuite ces travaux, les rangeait sur disque, puis les exécutait au moment jugé opportun. Le lot (batch) est donc bien l'unité de soumission dans ce type de système. Une fois le travail terminé, les résultats de toutes les étapes étaient regroupés et imprimés sous formes d'un listing. Un opérateur déliassait les listings des différents travaux, les

remettait au guichet où le paquet de cartes initial était regroupé avec le listing correspondant. L'utilisateur pouvait alors passer au guichet et y retirer l'ensemble.

D'où le terme de temps **différé**: le travail est réellement exécuté par le système informatique en dehors de la présence de l'utilisateur. Le temps de traitement correspond alors à la somme des temps suivants:
- temps de constitution d'un lot,
- temps de lecture, exécution et impression par l'ordinateur,
- temps dû au déliassage,
- temps de constitution de l'ensemble travail plus listing.

Ce mode de soumission a bien entendu évolué et l'on peut aussi soumettre des travaux stockés sur bandes magnétiques (bande IMCV sous GCOS3), sur disque, créés par d'autres travaux ou transactions (cf. chapitre 10), depuis un terminal interactif (écran/clavier).

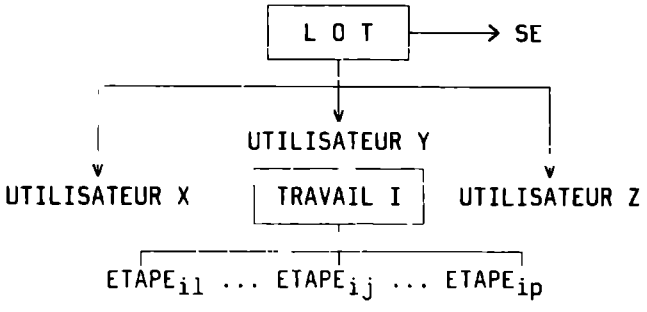


Figure 3d: relations entre utilisateurs et requêtes dans le traitement par lots.

3.3 – SE orienté vers le mode dialogué.

Ces SE sont plutôt conçus pour des environnements de **développement**, d'informatique déjà plus **individualisée**, voire départementalisée. Alors que les gros ordinateurs ont des SE dont la base est le traitement par lots, on trouvera plutôt les SE basés sur le mode dialogué dans les petits et moyens ordinateurs.

Ils sont conçus pour offrir des services **directs**, **agréables** (user friendly, ease of use, sont les expressions correspondantes du langage américain) à leurs utilisateurs. L'aspect ergonomique fait l'objet de recherches parfois poussées.

L'objectif primordial n'est plus l'utilisation globale des ressources, mais le **temps de réponse** (response time) qui est généralement défini comme le temps qui s'écoule entre le moment où l'utilisateur envoie sa requête (appui sur la touche "transmission" par exemple) et celui où le premier élément de la réponse émise par l'ordinateur s'affiche sur le poste de travail.

Vue du SE, la **charge** de travail (workload) qu'il va devoir écouler est a priori **inconnue**, et il devra s'y adapter **dynamiquement**.

Les requêtes émises par les utilisateurs se présentent sous forme de **commandes** (command) ou de réponses à des **menus** (menu-driven human interface, mode question/réponse). Contrairement au mode différé, ces commandes ne sont pas regroupées au préalable, mais émises une par une, selon les désirs des utilisateurs, et non pas selon les décisions

du SE. Elles sont cependant fonctionnellement équivalentes à celles du mode différé. Sur certains SE, la forme peut même être identique dans les modes différés et dialogués (cas de NOS de CDC).

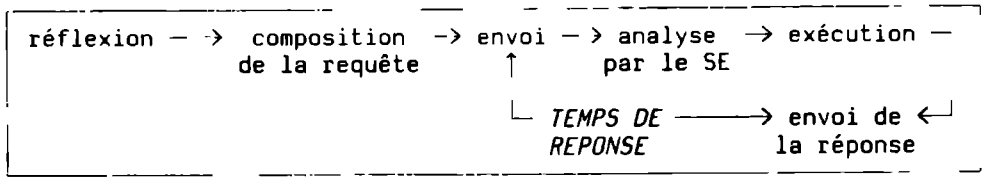


Figure 3e: requêtes dans le mode dialogué.

L'analogue du **travail** est ici la **session**, c'est-à-dire le temps écoulé entre les deux commandes particulières suivantes:
- la commande de **connexion** (login, logon, signon, hello, etc.) par laquelle l'utilisateur s'identifie auprès du SE et lui indique qu'il fait appel à ses services, après que le SE lui ait envoyé sa bannière (sa "mire");
- la commande de **déconnexion** (logoff, logout, signoff, bye, etc.) qui permet à l'utilisateur d'indiquer au système d'exploitation qu'il n'a plus besoin de ses services.

L'analogue de l'**étape** est la **commande** qui va provoquer l'exécution de la forme exécutable (cf. tome V) de cette commande.

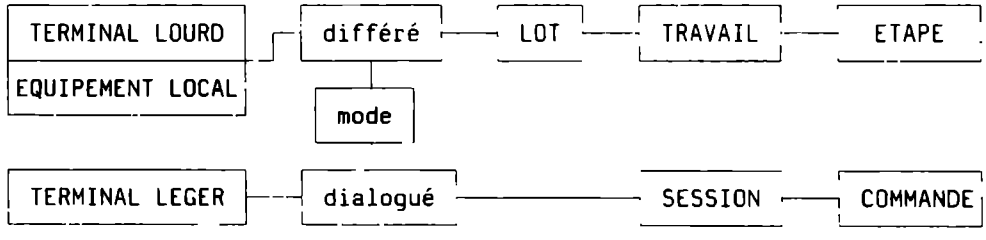


Figure 3f: équivalences entre requêtes et équipements selon les modes.

Sur le plan matériel l'utilisateur dispose d'un poste de travail individuel appelé **terminal**, qui se présente de nos jours sous forme d'un écran/clavier. A l'origine ce terminal ressemblait à une machine à écrire (MAE), technologie dont il était issu. L'un des modèles de clavier/imprimante les plus populaires fut l'ASR33 de la marque Teletype, souvent appelé TTY.

Ce terminal peut être relié directement par un câble à l'ordinateur: on le dira alors **local**. Lorsqu'il est relié par l'intermédiaire de liaisons téléphoniques, on parlera de terminal **distant** (remote).

Un autre critère conduit à le traiter de terminal **léger** par opposition aux terminaux **lourds** (work station) utilisés en mode différé pour l'entrée des travaux à distance: ces terminaux lourds correspon-

dent au déport des équipements locaux (lecteur de cartes, imprimante, dérouleur de bandes) servant à lire les travaux et à imprimer les résultats.

Une analogie avec les modes de communication entre humains conduit à associer l'interactif au téléphone, mode de dialogue synchrone où les interlocuteurs dialoguent par échanges continus après avoir établi une connexion, une session, et à associer le différé au courrier postal où les interlocuteurs dialoguent de façon asynchrone sans établir au préalable un chemin certifié de bout en bout.

Le mode dialogué, ou interactif (interactive), ou encore conversationnel, présente actuellement deux sous-modes:

- le temps partagé (time sharing), le plus ancien, qui tire son nom du fait que l'ordinateur sert simultanément plusieurs utilisateurs en partageant son temps entre eux;
- le transactionnel, plus récent. Contrairement au temps partagé où l'utilisateur dialogue directement avec le SE, l'utilisateur du transactionnel est un utilisateur final qui dialogue (en fait il répond à des questions posées par d'autres) avec des applications interactives auxquelles il a le droit d'accéder. C'est le cas, par exemple, des applications de réservation chez les transporteurs comme la SNCF, Air France, des applications d'interrogation de compte dans les banques, etc.

Dans le temps partagé l'utilisateur peut créer des objets comme les fichiers, lancer des programmes. Dans le transactionnel par contre l'utilisateur ne peut rien créer, mais remplir des cadres prédéfinis, des "grilles".

La tendance actuelle est à l'évolution vers les systèmes interactifs aussi bien sur les petits systèmes (domaine historique de l'interactif) que sur les gros systèmes.

CTSS (Compatible Time Sharing System) est un des plus anciens SE interactifs, développé au MIT au début des années 60 sur un IBM 7090 avec 64 Kmots de mémoire, 32 pour le SE, 32 pour les utilisateurs.

3.4 – Les SE orientés vers le temps réel.

Ces SE ressemblent à ceux du mode dialogué dans la mesure où ils servent des utilisateurs auxquels ils doivent répondre aussi vite que possible. Une des différences réside dans la vitesse de ces réponses: alors qu'elle est couramment de une ou plusieurs secondes dans le mode dialogué, elle est de quelques millisecondes dans les SE temps réel. Cette réponse doit obligatoirement arriver dans un laps de temps pour lequel il existe une valeur maximale définie en fonction des contraintes de fonctionnement et de sécurité. Le temps réel est moins "élastique" que le mode dialogué, il est plus "déterministe". A la limite le fait d'avoir à répondre avec un délai maximum déterminé à l'avance est une caractéristique plus discriminante que l'ordre de grandeur du temps de réponse.

Le pur SE temps réel (real time) ne dialogue pas véritablement avec de vrais utilisateurs, mais pilote des appareils, des processus industriels. Ceux-ci émettent des signaux, des données que l'ordinateur doit analyser très vite afin de réagir comme il convient.

Nous citerons comme exemples:
- le contrôle de l'état d'une pile atomique,
- les autocommutateurs, les commutateurs de paquets, les frontaux, les émulateurs de grappes de terminaux,
- le traitement d'image,
- le pilotage automatique d'un avion, d'une navette spatiale,
- la commande de robot.

La charge, c'est-à-dire la quantité de travail à traiter, est ici connue à l'avance, il serait sinon impossible d'y répondre avec les contraintes de temps requises.

	MODE DIFFERE	MODE DIALOGUE	TEMPS REEL
CHARGE	maîtrisable	inconnue	connue
TEMPS DE REPONSE	heures, jours	secondes	millisecondes
TERMINAL	lourd	léger	appareils
STIMULUS	lot	commande	signal

Figure 3g: les grandes caractéristiques des types de SE.

Une autre particularité des SE temps réel est le caractère **résident** des logiciels en mémoire centrale. Comme tout est prévu à l'avance et doit être exécuté aussi vite que possible, tous les logiciels sont chargés au démarrage du système. A la limite il n'y a pas de chargeur d'application (cas de CORTEX de CSEE par exemple), l'ajout d'un nouveau logiciel nécessitant une nouvelle "génération" du système. Les SE temps réel sont de plus généralement multitâches (cf. chapitre 10), et peuvent fréquemment être considérés comme mono-utilisateur, la notion d'utilisateur n'étant pas explicitement implémentée puisque l'ensemble des tâches forment une seule application. Des configurations temps réel tournent couramment sans disque (memory based systems) et on pourra encore y trouver des mémoires centrales à tore (mémoires magnétiques) qui présentent l'avantage de ne pas perdre l'information lorsque l'alimentation fait défaut. La mémoire virtuelle (chapitre 12) en est absente ou non utilisée. Il y a rarement présence d'un pupitreur, le système étant aussi automatique que possible.

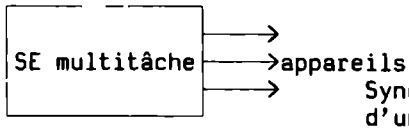


Figure 3h: environnement d'un SE temps réel.

Synonyme d'efficacité pour les performances, d'une certaine pauvreté fonctionnelle.

Les **interruptions** (cf. tome III), nombreuses et hiérarchisées constituent une autre caractéristique des ordinateurs dotés de ce type de SE. Le matériel doit être conçu pour "répondre" rapidement aux interruptions, c'est-à-dire activer la tâche associée: l'iAPX286 est ainsi présenté comme étant capable de répondre en moins de quatre microsecondes. Les interruptions sont associées à une gestion des requêtes basées sur des priorités fixes.

Ce sont généralement des mini-ordinateurs et de plus en plus des multimicros (cf. tome I, chapitre 20). Leurs outils de développement sont frustes dans de nombreux cas, à tel point que certains font appel à un autre SE pour le développement.

La disponibilité élevée est une autre caractéristique de ces systèmes et l'on trouvera fréquemment des systèmes redondants (cf. tome I, chapitre 19). On trouvera dans [RAU86] une étude sur Unix et le temps réel avec un résumé des caractéristiques principales des SE temps réel, telles que les définissent les comités ou groupes de l'IEEE et de /usr/group travaillant sur ce sujet:

- aptitude à répondre à des événements asynchrones;
- temps de commutation de contexte (de processus) minimum;
- temps minimum de prise en compte des interruptions;
- interruptions avec priorités et distribution (dispatching) basée sur des priorités;
- distribution pré-emptive;
- fixation en mémoire;
- fichiers contigus;
- synchronisation.

MODE DIFFERE	MODE DIALOGUE		TEMPS REEL
	TEMPS PARTAGE	TRANSACTIONNEL	
GCOS3	MULTICS	GUARDIAN	RTE-A (HP1000)
GCOS8	PRIMOS	SSP (S/36)	CORTEX (CSEE)
GCOS7	UNIX	CPF (S/38)	iRMX86 (Intel)
MVS	GCOS6	ACP devenu TPF	ATHOS (X83)
DOS/VSE	MPE (HP3000)		ISIS (Tymshare)
MCP (Burroughs)	MS-DOS	EXEC 32000 (NAS)	RPS (Série 1)
OS1100	AOS/VS (DG)	AOS/TR 32	RT-11 (DEC)
NOS (CDC)	CP/M	DRM (Philips)	UNIX RTR (MERT)
George 3 (ICL)	VMS	SCEPTRE	RT VOS (Harris)
VME/B (ICL)	STARSYS (CTOS)	VAX ELN	MPX-32 (Gould)
	AMIGADOS	MAX (Modcomp)	RMS 68K
	DX10 (TI)	RTES (SPS5)	MP/AOS (DG,16b)
	VULCAN	VRTX	R-CRAFT (Tecsi)
	TOPS (DEC)	pSOS (68000)	SPART (SPS7)
	METEOR	SMRT (SPS7)	MMOS (FLEX/32)

Tableau 3i: exemples de systèmes d'exploitation.

Si le partage des ressources est un élément de base dans la conception des SE orientés vers le temps réel, il est presque antinomique avec les SE temps réel puisque ceux-ci ont besoin de disposer des ressources nécessaires dès que l'information associée arrive. La différence entre les SE conçus pour l'interactif et ceux conçus pour le temps réel est telle que AT&T par exemple a développé un SE temps réel MERT (devenu DMERT) à partir d'Unix. Pour les SE temps réel on constate, à l'heure actuelle, une tendance à conserver leurs

spécificités, en incluant les caractéristiques des SE interactifs: ceci est typique de l'évolution de RTE-A sur HP1000. Tous les SE décrits dans la colonne "temps réel" du tableau 3h y figurent en fonction des définitions données par les vendeurs, certains ne devraient pas y figurer selon nous. Viser le temps partagé et le temps réel simultanément n'est pas une opération sans risque: VAX/VMS s'est ainsi imposé en temps partagé, mais sans remplacer les PDP-11 en environnement temps réel, malgré le mode compatibilité du VAX et la volonté de DEC.

3.5 – SE et tailles des ordinateurs.

Le classement précédent est basé sur le mode de dialogue entre l'utilisateur et l'ordinateur. On peut aussi classer les SE en fonction de la taille des ordinateurs auxquels ils sont destinés.

Il est en effet très difficile, sinon impossible d'avoir un SE couvrant tout le spectre d'une gamme d'ordinateurs. Un SE trop puissant fonctionnellement aura tendance à "écraser" un ordinateur moyen qui lui fournira tout juste suffisamment de ressources pour s'exécuter lui-même. Il en est de même pour une grosse voiture avec un petit moteur et l'on constate que les constructeurs adaptent la taille de la carrosserie à la puisssance du groupe moteur. Dans le domaine de la gestion il est ainsi courant de trouver au moins deux SE, l'un pour le haut de la gamme, l'autre pour le milieu de la gamme:

- dans les 370 d'IBM on a eu ainsi les OS (Operating System) pour le haut de gamme; MVS en est le principal représentant actuel. DOS (Disk Operating System) est utilisé en milieu et bas de gamme (sur les 43XX actuellement). IBM a même proposé plus récemment une version réduite du DOS, appelée SSX (Small System executive) pour le très bas de gamme;
- chez Bull GCOS3 puis GCOS8 habillent le haut de gamme, alors que GCOS7 est fourni sur le milieu de gamme. Dans le cas de Bull, les matériels relèvent en plus d'architectures externes (cf. tome I) différentes.

Les SE ont parfois le même nom sur les différents modèles de certains constructeurs, alors qu'ils sont tout à fait différents (cas des MCP de Burroughs).

3.6 – SE et architecture externe.

Un SE est généralement conçu pour une architecture externe et une seule: il n'est pas question que MVS d'IBM s'exécute sur autre chose qu'un processeur déroulant du code compatible avec celui des 370.

Unix constitue une exception: sa concision, sa puissance et le fait qu'il est écrit dans un langage évolué font qu'il a été installé aussi bien sur plusieurs micro-ordinateurs que sur de très gros systèmes scientifiques comme le CRAY 2 ou de gestion comme UTS d'Amdahl. P-System et Pick constituent d'autres exceptions.

Les micro-ordinateurs ont aussi été dotés de SE qui n'étaient pas écrits par des fabricants d'ordinateurs. Bien que quelques-uns aient été implantés sur plusieurs microprocesseurs, chacun a cependant un microprocesseur, ou une famille de microprocesseurs, privilégié.

chapitre 4

structure générale d'un système d'exploitation

Nous avons déjà abordé la structure générale du logiciel dans le chapitre 1 et fait apparaître le système d'exploitation. Dans ce chapitre nous donnons la structure générale des systèmes d'exploitation en analysant la façon dont les requêtes des utilisateurs les traversent. Par "requête", il faut entendre "travail", "commande".

Nous utiliserons de préférence les SE orientés vers le traitement par lots (mode différé) car ils permettent de mieux dégager cette structure générale, tout particulièrement pour l'activité de planification.

4.1 – Les trois phases d'une requête.

Le traitement de la requête d'un utilisateur parcourt trois phases successives:

- dans une première phase elle est **lue** par le SE, stockée, et reste en attente;
- après décision du SE, la requête passe dans la phase d'**exécution**, phase dans laquelle le code de l'utilisateur est déroulé;
- une fois son exécution terminée, la requête passe dans la phase de **terminaison**, ses résultats sont imprimés et toute trace de son passage est effacée. Elle fera l'objet d'une "purge".

Le système d'exploitation va être structuré en fonction de ces trois phases et comprendra trois sous-ensembles:

- le **sous-système d'entrée** des requêtes:
  * entrée des travaux dans un système orienté lot. JES comme Job Entry Subsystem sous les OS d'IBM, POWER sous les DOS d'IBM, GIN comme General INput sous GCOS3;
  * connexion des utilisateurs dans le cas d'un système interactif. Dans Multics le processus "initializer" gère l'entrée des utilisateurs (login), dans le cas de GCOS6-MOD400 on parlera du "listener". Sous RTE-A (HP1000) avec l'option VC+, PROMPT joue ce rôle en traitant les interruptions "non sollicitées".

  Le sous-système d'entrée a pour rôle:
  * la **lecture** des requêtes,
  * l'**analyse** de ces requêtes afin de vérifier si leur syntaxe est

correcte, si l'utilisateur a le droit d'accéder aux ressources demandées,
  * la **planification** des requêtes (scheduling);
- l'**allocation** des ressources nécessaires à la bonne exécution des requêtes correspond à la phase d'exécution. C'est le rôle des allocataires (allocator) de ressources comme le distributeur (dispatcher) au niveau du processeur central;
- le **sous-système de sortie** chargé d'imprimer les résultats et de purger le système des traces du passage de la requête. GOT (General OUtput) joue ce rôle dans GCOS3, JES dans MVS.

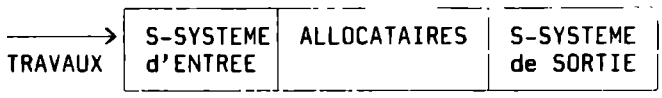


Figure 4a: structure générale d'un SE.

4.2 – Le système d'entrée des requêtes.

4.2.1 – Lecture des requêtes.

Les requêtes sont lues sur le périphérique d'entrée des travaux (historiquement un lecteur de cartes). Ce périphérique physique est ensuite devenu un objet de nature logique, un fichier d'entrée, couramment appelé SYSIN (comme SYStem INput) pour la majeure partie du logiciel, hormis les composants du SE chargés de l'alimenter physiquement depuis les périphériques réels.

Sous GCOS3 on parlera pour le SYSIN du fichier *J. On parlera de "lecteur interne" (internal reader) dans JES pour des lecteurs qui ne correspondent pas à des équipements physiques, mais permettent à des programmes de soumettre des requêtes sans passer par la périphérie. Sous OS d'IBM, les "input readers" autrefois chargés d'alimenter le fichier SYSIN, autrefois incorporés dans le SE, font maintenant partie de JES. Sous GCOS7 le terme de "input reader" correspond à une notion différente illustrée figure 4c.

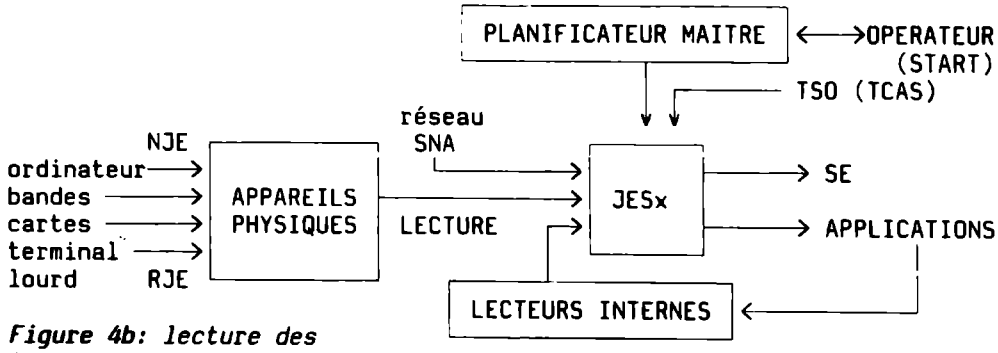


Figure 4b: lecture des travaux dans un SE du type MVS.

Une fois le travail lu, le système d'entrée écrit un article dans le fichier comptable (SMF sous MVS, SCF sous GCOS3) avec au minimum:
- les date et heure de la lecture,
- l'identification de la requête,
- le nombre de cartes ou d'images de cartes lues.

Ces informations pourront être traitées plus tard afin de facturer les utilisateurs, de dresser des statistiques sur l'utilisation du système informatique, sur les temps de réponse, sur la typologie des requêtes.

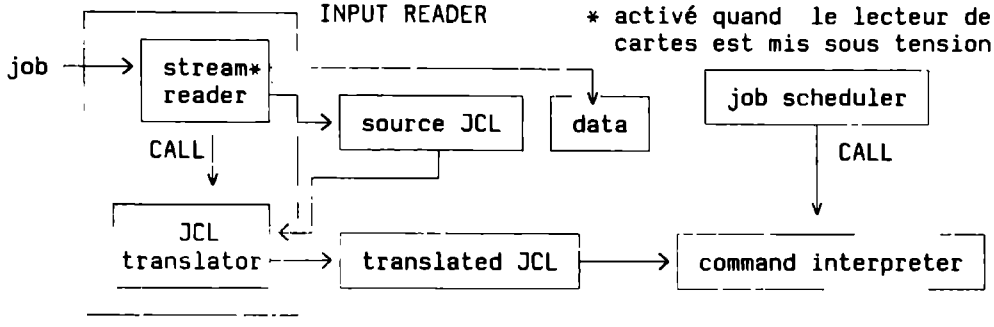


Figure 4c: la notion de "input reader" sur GCOS7.

Les travaux ainsi lus sont stockés dans des fichiers dédiés au SE: la SYS1.SYSJOBQE (comme "job queue", file des travaux en attente) des anciens OS d'IBM est remplacée par les SWA et les spoules de JES dans MVS; la contention, donc les pertes de temps causées par la sérialisation des requêtes à une ressource unique, diminue. Avec MS-DOS, les fichiers standard zéro et un (clavier STDIN et écran STDOUT par défaut) jouent le rôle de SYSIN et SYSOUT.

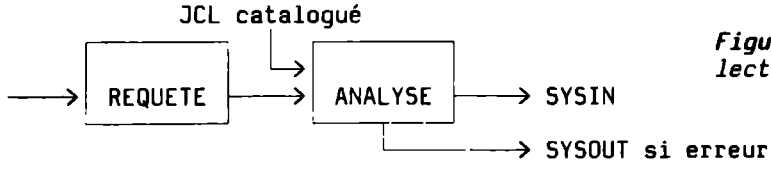


Figure 4d: phases de lecture et analyse.

4.2.2 – Analyse.

Durant la lecture, ou après lecture et stockage vus ci-dessus, les commandes trouvées dans le flot entrant (cartes commençant par // sous OS d'IBM, par $ sous GCOS) sont analysées et converties en format interne. En cas d'erreur, la requête est rejetée et mise dans la file d'attente du système de sortie (figure 4d). S'il n'y a pas d'erreur, la requête se voit attribuer une identification ("job id" dans JES) qui permettra de la référencer de façon non ambigüe au cours de son exécution. C'est durant cette phase que peuvent être prises en compte les procédures cataloguées ("// EXEC PROC=" dans DOS/VSE, celles de la librairie SYS1.PROCLIB sous JES, cf. chapitre 15). La commande

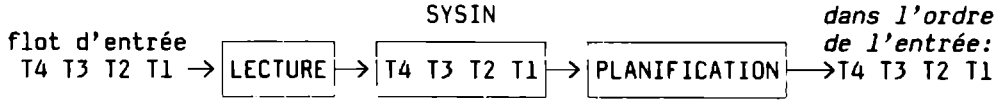
$ SELECT de GCOS3 permet aussi de fusionner avec le JCL soumis du JCL pré-stocké dans des fichiers.

Dans le cas d'un SE interactif (orienté vers le mode dialogué) ces deux phases de lecture et d'analyse se limitent à la vérification des droits d'accès de l'utilisateur: vérification du mot de passe, du numéro du terminal utilisé parfois (un des rôles du "login externe" de Primos). Les phases de lecture et d'analyse se reproduisent pour chacune des commandes, au moment où elles sont frappées. Afin d'éviter toute ambiguïté, précisons pour les informaticiens avertis que les sous-systèmes en temps partagé comme TSO ou TSS ne sont pas considérés comme des SE interactifs.

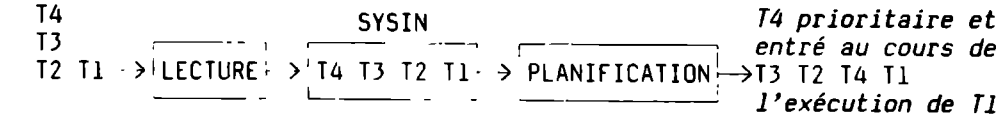
4.2.3 – Planification.

La planification consiste à choisir une requête parmi celles en attente, afin de la faire passer réellement en exécution. Ce choix peut être très simple et se réduire à vérifier que le nombre maximum d'utilisateurs simultanément admis n'est pas dépassé, cas que l'on trouve sur de nombreux SE interactifs. Ce choix peut aussi relever d'algorithmes complexes. Selon les SE on trouvera les mécanismes suivants:

- planification **séquentielle**: les travaux sont planifiés dans l'ordre de leur lecture (FCFS comme First Come First Served, premier arrivé, premier servi). Dans un système d'exploitation à monoprogrammation (cf. chapitre 10) cette planification correspondra à un algorithme de type FIFO (First In First Out, premier entré, premier sorti):



- planification basée sur des **priorités** numériques: chaque requête a une priorité de 1 à n. La requête la plus prioritaire est servie en premier. Les requêtes qui attendent trop longtemps peuvent voir leur priorité évoluer de façon dynamique dans certains systèmes: "overdue" de GCOS3, "passover limit", "priority aging" de JES3 et JES2. JES2 augmente la priorité dans la classe en fonction du temps, JES3 l'augmente chaque fois que la requête est évaluée sans être planifiée (paramètres MAGEL, SAGEL, MAGER, SAGER de la commande SELECT);

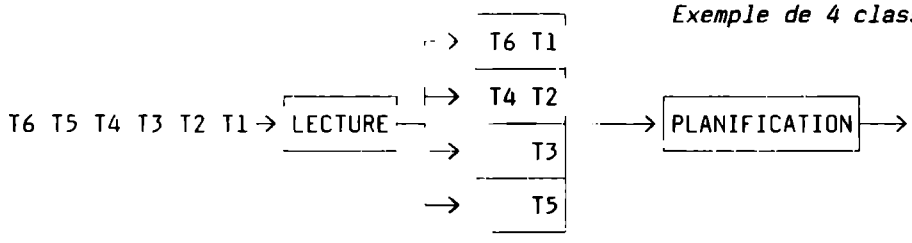


- planification basée sur des **classes** de requêtes: chaque requête est rangée dans une classe:
  * selon un paramètre: c'est le cas de JES avec le paramètre CLASS dans la commande // JOB, de la commande //*MAIN avec JES3,

* en fonction de l'analyse de la structure de la requête: cas de GCOS3 où le module "Job Classifier" choisit la classe dans laquelle une requête est rangée, le "Job Dispenser" choisissant la requête suivante à exécuter.

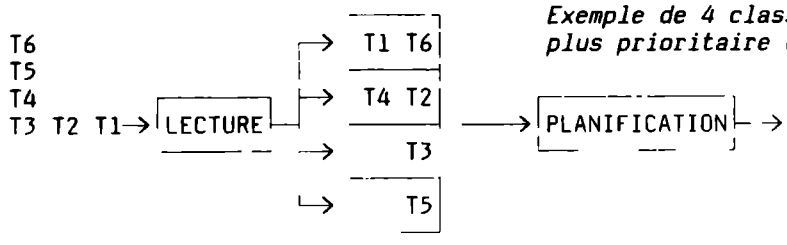
Le but est d'exécuter en parallèle des travaux complémentaires: l'un chargera le processeur, l'autre la mémoire, un autre les bandes, etc. (cf. paramètre CHOICE JOBMIX de la commande SELECT de JES3). GCOS-64-E permet ainsi 16 classes pour les requêtes des utilisateurs, plus 6 pour celles du système lui-même, JES3 offre 256 classes sur des machines plus puissantes. A un moment donné il y a un nombre précis de requêtes de chaque classe en cours d'exécution (cf. paramètres MDEPTH, MLIMIT, TDEPTH, TLIMIT de la commande CLASS de JES3), et chaque fois qu'une requête se termine, une requête de la même classe peut être choisie. L'algorithme d'analyse des classes peut encore favoriser les classes pour lesquelles aucun travail n'a été sélectionné depuis longtemps (cas de JES);

Exemple de 4 classes.



- planification utilisant priorités et classes: à l'intérieur de chaque classe, les requêtes sont planifiées en fonction de leur priorité. C'est le rôle des paramètres CLASS et PRTY avec JES, du paramètre urgence de la commande $ SNUMB de GCOS3;

Exemple de 4 classes, T6 étant plus prioritaire que T1.



- planification par l'**heure de début** d'exécution: la requête doit être lancée tel jour à telle heure. C'est le rôle de la commande $ MSG3 sous GCOS3 où les requêtes ainsi mises en attente vont dans la classe HOLD, de la commande "absentee" avec l'option "differed" de Multics, de la commande AT d'Unix et RTE-A;
- planification en fonction de l'**heure de fin** d'exécution (dead line scheduling). Cette méthode peut se révéler néfaste pour la bonne utilisation globale des ressources: plus le moment de la fin de l'exécution approche, plus le SE aura tendance à consacrer le maximum de ressources aux requêtes ainsi planifiées. GCOS3 n'offre pas

ce type de planification, JES3 en dispose: la commande

//*MAIN DEADLINE=(1000,B)

permettra à l'utilisateur de demander que son programme se termine à 10 heures en utilisant l'algorithme B (JES3 supporte jusqu'à 36 algorithmes différents pour ce type de planification) qui est défini dans l'"Inish Deck" par

DEADLINE,B=(+3,2H,+1,30M)

ce qui signifie que deux heures avant la fin de l'exécution demandée par l'utilisateur, il faudra incrémenter la priorité de 3; il faudra ensuite l'augmenter de 1 toutes les 30 minutes ([ROG85]);

- planification en fonction de l'utilisation présumée des ressources (projected resource utilization): l'utilisateur indique dans les commandes les caractéristiques de sa requête:
  * temps processeur central: $ LIMITS sous GCOS3, paramètre TIME des commandes // JOB ou // EXEC des OS d'IBM;
  * nombre de lignes imprimées: $ LIMITS de GCOS3, paramètre OUTLIM dans les commandes // DD sur OS;
  * taille mémoire: commande $ LIMITS de GCOS3, paramètre REGION chez IBM;
  * temps et/ou nombre d'entrées/sorties (paramètres IORATE de la commande //*MAIN de JES3);
  * place sur disque: paramètre SPACE dans les cartes // DD chez IBM;
  * nombre de dérouleurs;
  * nombre d'imprimantes dédiées, etc.

  Si l'une de ces limites est dépassée, la requête est stoppée avant sa terminaison normale (abort, cancel). Sur GCOS3 c'est maintenant au pupitreur de "tuer" ou laisser continuer le travail en cas de dépassement des limites; les "petites" requêtes (classe "express") y sont planifiées en priorité car on estime qu'il y aura toujours de quoi les exécuter. Une fonctionnalité analogue a été ajoutée à JES2: "exec batch scheduling" dans lequel JES2 ne filtre plus les commandes, les travaux correspondants deviennent des commandes directes de l'initiateur; cette fonction a été conçue initialement pour le monde universitaire où les étudiants font beaucoup de travaux du type CLG (Compile, Link, Go, c'est-à-dire compilation, édition de liens et exécution dans la foulée) selon [ROG85];
- planification en fonction d'autres requêtes (DJC comme "dependent job scheduling" de JES3) qui doivent avoir été exécutés préalablement;
- planification en fonction de contrats de service passés entre les utilisateurs et le service informatique: un groupe d'utilisateurs se verra garantir un certain taux d'utilisation des ressources.

Ces mécanismes peuvent être mis en oeuvre de façon cyclique: une requête sera ainsi soumise toutes les heures, tous les jours, tous les dix jours, etc.

Les algorithmes ci-dessus ne tiennent pas compte de la charge réelle de l'ordinateur et peuvent conduire à mettre en exécution des requêtes qui se gênent mutuellement parce qu'elles saturent la mémoire centrale par exemple. Par charge réelle nous entendons l'ensemble des travaux planifiés, ceux qui sont en cours d'exécution et se voient

allouer des ressources, par opposition à la charge soumise qui comprend tous les travaux en attente dans le sous-système d'entrée. La prise en compte de la charge par le biais du nombre de travaux en cours d'exécution est une approche trop grossière car elle n'est pas liée au comportement instantané des travaux. Les planificateurs des SE modernes prennent en compte de plus en plus finement l'état instantané de l'ordinateur. Le paramètre MAXUSER de MVS permet ainsi de limiter le nombre de requêtes ("d'espaces d'adresses", donc la taille maximum de l'espace occupé par les pages en mémoire virtuelle) simultanément actives, et surtout SRM (composante "resource management") adaptera dynamiquement le nombre de travaux actifs ("swapped in"), donc le taux de multiprogrammation, en fonction de la charge (cf. chapitre 11).

Précisons que dans MVS il y a planification lorsqu'un initiateur est disponible. Il choisira un travail dans une des classes auxquelles il est affecté. Dans GCOS3, la planification est réalisée par un travail (au sens GCOS) appelé SCHEDULER.

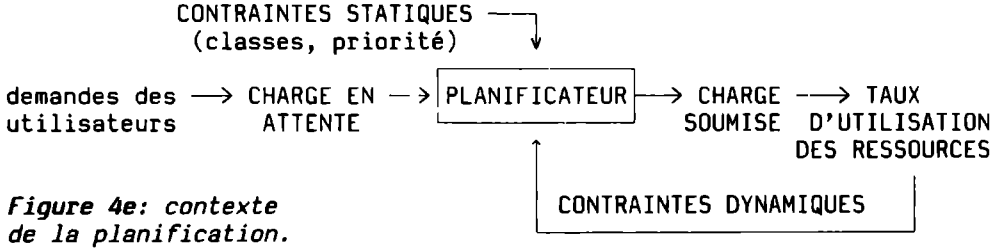
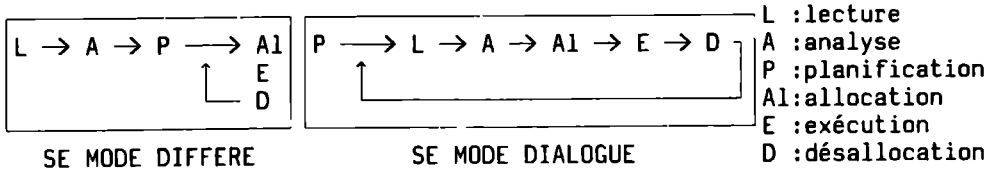


Figure 4e: contexte de la planification.

L'ordonnancement qui élargit le cadre de la planification des travaux aux applications de l'entreprise n'est pas traité ici, car il déborde du cadre du système d'exploitation. JES2 avec le produit programme OPC (Operations Planning & Control) offre des fonctions d'ordonnancement.

4.2.4 – Cas des SE interactifs.

Nous avons jusqu'ici orienté la présentation de la structure des SE en utilisant plutôt le mode différé. Le problème de la planification se présente sous deux aspects dans les SE orientés vers le mode dialogué.



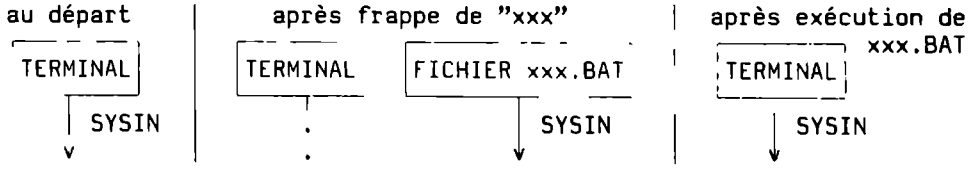
Le premier aspect est lié à l'analogie entre travail et session. La planification peut se réduire à sa plus simple expression ou faire intervenir des contraintes liées aux contrats de service, à la charge

instantanée du système, etc. Le "load control group" (groupe de contrôle de la charge), la commande MAXU de Multics se situent dans cette perspective. Sous VMS, la commande SET LOGINS/INTERACTIVE (nombre d'utilisateurs connectés), les paramètres MAXPROCESSCNT (nombre de processus), RJOBLIM (nombre de terminaux distants), l'utilitaire d'autorisation (Authorize Utility) définissant les heures permises à chaque utilisateur, vont dans le même sens.

Le second aspect provient d'une forme simulée de traitement par lots: l'utilisateur interactif peut soumettre des "travaux", c'est-à-dire des fichiers contenant des commandes (JCL). Ces travaux se dérouleront en parallèle avec les commandes interactives frappées sur le terminal qui les a soumis au SE. A un moment donné il y aura un seul travail "batch" par file (queue) simulée. Les travaux "batch" seront en quelque sorte des sessions sans terminal. A la différence des SE orientés vers le traitement par lots, les SE interactifs n'analysent pas globalement la requête au moment de leur soumission. S'il y a une erreur dans l'écriture de la vingtième commande, elle ne sera détectée qu'après exécution des dix-neuf premières commandes.

Le fichier d'entrée (SYSIN) est par défaut le terminal mais peut généralement être modifié: "redirection" d'Unix ou MS-DOS avec le caractère "<", redirection des LU (Logical Unit) sous RTE-A. Cette modification du fichier d'entrée permet d'ailleurs de soumettre les requêtes sous plusieurs formes:

- un fichier contenant des commandes deviendra le nouveau fichier SYSIN, de façon temporaire. Cas des fichiers de type "bat" (AUTOEXEC.BAT par exemple) de MS-DOS, cas des fichiers de type "com" ou "cmd" de VMS lancés par la commande "@", etc. Le terminal est alors "bloqué": le SE ne prendra en compte la commande suivante que lorsque les commandes du fichier "SYSIN" seront terminées. Dans MS-DOS version 2.0 ceci est généralisé par les unités standard (mécanisme de "handle"), l'unité zéro étant l'entrée standard, STDIN, redirigeable (par "<") vers un fichier:



- le fichier de commandes soumis dans une file "batch" devient fichier SYSIN pendant le déroulement de la requête: cas de RUN/QUEUE=XXX de VMS, cas des "absentee" de Multics, de la commande JOB de Primos. Contrairement au cas précédent, le terminal d'où la requête a été lancée n'est pas bloqué par le déroulement de cette requête;
- le même fichier de commandes peut encore être lancé, dans certains SE, sans bloquer le terminal, sans entrer dans une file "batch". Il se déroule en parallèle avec les commandes frappées sur le terminal: cas des "phantoms" de Primos, des DRUN de l'environnement interactif de GCOS3.

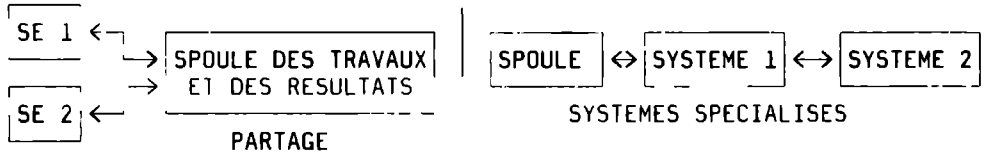
Dans les contextes interactifs on qualifiera d'**avant-plan** (foreground) une requête qui s'exécute de façon synchrone (en bloquant le terminal), et d'arrière-plan (background) une requête qui se déroule de façon **asynchrone** par rapport au terminal d'où elle est issue. Unix permet d'ailleurs d'exécuter toute commande dans l'un des deux modes, par la simple utilisation du caractère "&" en fin de commande. La commande BACK d'iRMX joue le même rôle que "&" dans Unix.

Quand nous parlons de bloquer le terminal, nous entendons que le SE n'exécutera pas d'autre requête avant la terminaison de la requête en cours. Ceci ne signifie nullement qu'il est impossible de frapper de nouvelles requêtes avant la terminaison des précédentes: de nombreux SE (Multics, MS-DOS, iRMX86, Primos, VMS, etc.) permettent la frappe anticipée (type ahead).

4.2.5 – Cas des multisystèmes.

Les multisystèmes (cf. tome I) interviennent de deux façons dans l'entrée et la sortie des travaux selon que:

- certains systèmes sont **spécialisés** dans la fonction d'entrée et sortie: cas des 7040 d'IBM dans la configuration DCS;
- tous les systèmes se **partagent** les mêmes files d'entrée (SYSIN, spoule d'entrée) et sortie (SYSOUT, spoule de sortie) des travaux.



Partage des entrées et sorties des travaux par plusieurs systèmes d'exploitation et systèmes spécialisés.

Dans le cas du partage, chaque ordinateur a sa copie du SE et l'on distingue:

- la gestion **centralisée** de la file des travaux, cas de JES3, évolution de ASP qui date de 1968 et était lui-même initialement inspiré de DCS;
- la gestion décentralisée, chaque sous-système d'entrée pouvant choisir directement un travail dans la file d'attente (cas de SMS de GCOS3, de JES2).

Ce partage de la file des travaux en attente permet de présenter à l'utilisateur une puissance de traitement plus importante (constituée de plusieurs ordinateurs), il permet aussi de mieux utiliser l'ensemble des ressources puisque les travaux peuvent être dirigés sur l'ordinateur le moins chargé, ce qui correspond à l'**équilibrage de charge** (load balancing). Ce partage offre aussi une meilleure résistance du service face aux pannes d'un des ordinateurs. Nous verrons plus loin que ce souci a été étendu aux ensembles formés d'ordinateurs reliés par des réseaux, et l'on parlera alors d'entrée des travaux

dans le réseau (cf. l'architecture NJE, Network Job Entry, avec les produits NJP extension de JES3, RSCS sous VM).

Le partage du spoule sous GCOS3 est appelé SMS (Shared Mass Storage), la synchronisation des accès étant réalisée par des verrous (gate, cf. chapitre 14) gérés par les contrôleurs de disque. Le VAXCluster qui regroupe plusieurs machines VMS offre aussi une file d'entrée/sortie commune aux travaux (system-wide queues, generic queues pour l'équilibrage de charge).

Dans quelques systèmes un ordinateur est spécialisé dans la gestion des périphériques lents et des lignes de communication. Ce fut le cas de DCS (Direct Coupled System) avec un IBM 704X en frontal d'un 709X reliés par canal (CTC).

4.3 – Allocation des ressources.

Une fois la requête planifiée, il va falloir lui attribuer les ressources nécessaires à son exécution. Dans un SE orienté vers le traitement par lots, les ressources ne sont pas attribuées au travail, mais à une étape de ce travail, l'étape à exécuter.

Le SE commence donc par attribuer les ressources demandées par la première étape. Une fois cette étape terminée, les allocateurs de ressource interviendront à nouveau pour attribuer les ressources nécessaires à la seconde étape, et ainsi de suite.

Le travail apparaît donc comme l'**unité de planification**, l'**étape** est l'**unité d'allocation** des ressources. L'exécution d'un travail revient ainsi à enchaîner plusieurs fois le cycle:

allocation – exécution – récupération

Les fonctions liées à l'allocation des périphériques sont de:
- localiser les fichiers,
- régler les problèmes de relations entre plusieurs requêtes d'allocation (concurrence d'accès),
- créer de nouveaux fichiers en utilisant les services du gestionnaire de fichier (SGF),
- affecter les périphériques aux requêtes,
- demander à l'opérateur de monter les volumes requis,
- permettre la concaténation dynamique des fichiers (rôle de l'initiateur sous MVS, JES ne le fait que pour les PROCLIB).

Le responsable de l'attribution du processeur est le **distributeur** (dispatcher) ou répartiteur.

Sous MVS, ce sont les "initiateurs" (initiator) qui sont responsables de l'allocation des ressources périphériques avec JES2: l'initiateur (un programme système lancé par un opérateur et/ou par JES2) demande à JES2 le prochain travail planifié. Il appelle ensuite l'interpréteur qui, à partir de la forme interne, construit les blocs de contrôle depuis la SWA (System Work Area) dans la zone de communication. Les ressources sont alors attribuées, puis l'initiateur lance le programme défini dans la commande // EXEC.

JES3 peut contrôler l'allocation (statique) de ressources périphériques en évitant la planification de travaux nécessitant ces

ressources, tant que les travaux qui les possèdent ne sont pas terminés. L'objectif est d'éviter le vidage (swap-out) des travaux, au prix d'une double allocation, la seconde par l'initiateur étant alors assurée d'être réussie immédiatement. En fait un périphérique est défini dans ces systèmes comme étant:
- géré exclusivement par JES3 (rôle de MDS, Main Device Scheduler),
- géré conjointement par JES3 et MVS,
- géré uniquement par MVS.

Ceci est dû au fait que MVS est conçu pour gérer un ordinateur alors que JES3 porte sur un ensemble (un "complexe") d'ordinateurs reliés par une liaison canal à canal (CTC, voir tome III). Deux copies de MVS pourraient allouer le même fichier à deux travaux qui le modifient, avec des résultats incohérents.

Sous GCOS3 c'est PALC (Peripheral ALloCator) qui est responsable de l'attribution des ressources périphériques, CALC (Core ALloCator) est lui responsable de l'attribution de la mémoire centrale. Sous Multics le RCP (Resource Control Package) alloue les ressources périphériques.

4.3.1 – Ordre d'allocation.

Les ressources sont généralement attribuées selon un ordre qui va des ressources les moins coûteuses aux plus coûteuses. Cette stratégie est logique puisqu'entre l'allocation de la première ressource et celle de la dernière ressource demandée par une étape il se déroulera un certain temps. Mieux vaut donc ne pas prendre le risque de "geler" des ressources coûteuses. Ce temps pourra être cependant minimisé en augmentant la priorité de planification au fur et à mesure de l'allocation des nouvelles ressources (paramètre INCR de la commande SELECT de JES3). On attribuera ainsi les **ressources périphériques** en premier:
- les unités **non partageables** d'abord:
  * les imprimantes,
  * les lecteurs de cartes,
  * les perforateurs de cartes,
  * les lecteurs de cassettes,
  * les lecteurs de disquettes considérés comme des périphériques lents et non partageables entre plusieurs programmes en dehors des micro-ordinateurs,
  * les dérouleurs de bande magnétique,
  * les disques privés,
- les unités **partageables** ensuite: disques, tambours.

Les ressources **centrales** sont attribuées en dernier:
- mémoire centrale dans un premier temps,
- processeur central avec ses registres ensuite,
- processeurs d'entrée/sortie, canaux, chemins vers la périphérie en dernier lieu, lorsque le processeur central a créé des entrées-sorties.

Nous retrouvons ici la hiérarchie des mémoires que nous avons introduite dans le tome I.

Dans certains SE, comme OS/MFT d'IBM en particulier, on a pu attribuer la mémoire centrale avant les ressources périphériques. Cette stratégie est liée à un type de gestion statique de la mémoire centrale que nous verrons plus loin (chapitre 11).

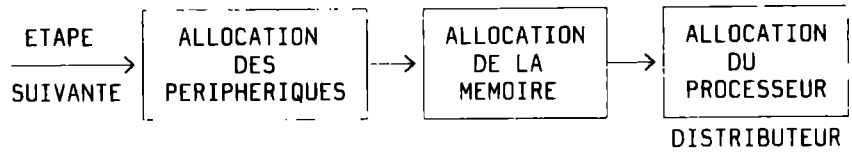


Figure 4f: ordre d'allocation des ressources.

4.3.2 – Allocations statique et dynamique.

La méthode la plus simple pour connaître les ressources dont a besoin une étape est que l'étape les déclare. Le SE analyse ces déclarations et attribue toutes les ressources demandées s'il le peut. S'il ne peut attribuer toutes ces ressources, l'étape attend. Les systèmes d'exploitation orientés vers le traitement par lots ont généralement été conçus de cette façon, et les commandes trouvées dans le flot d'entrée (dites "cartes de contrôle" parce que l'on utilisait des cartes au début) indiquent toutes les ressources nécessaires: les commandes // DD indiquent ainsi les besoins en fichiers et périphériques dans les systèmes OS d'IBM, les commandes équivalentes de GCOS3 seront $ PRMFL pour les fichiers permanents, $ FILE pour les fichiers temporaires, $ TAPE pour les bandes, $ PRINT pour une imprimante, $ PUNCH pour un perforateur de cartes.

Ce type d'allocation sera dit **statique** puisque tout est alloué en début d'étape, pour toute la durée de l'étape. Les ressources sont bloquées par l'étape, même durant les instants où elles ne sont pas réellement utilisées, même si l'exécution de l'étape ne les utilise pas systématiquement: si l'étape n'utilise 250 Ko de mémoire que pendant un dixième de son temps d'exécution, il faudra malgré tout les lui attribuer durant son exécution complète. D'autres étapes d'autres travaux risquent donc d'attendre plus longtemps et il faut se rappeler que la bonne utilisation de l'ensemble des ressources est un critère de qualité des systèmes d'exploitation.

L'évolution des systèmes d'exploitation suit une progression continue vers une allocation de plus en plus **dynamique** des ressources. D'une part on repousse le moment de l'allocation jusqu'à l'instant où la ressource va être effectivement utilisée, d'autre part, pour en faciliter la gestion, on décompose les ressources en quantum:
- cadre pour la mémoire (cf. chapitre 11),
- secteur pour les disques (cf. tome IV),
- paquet pour les bus, les lignes de communication, etc.,

et la ressource est attribuée, au moment de l'utilisation, par quantités de n quanta: un groupe de secteurs sur disque (cluster, block) par exemple.

On pourra trouver, à l'opposé, une allocation de nature encore plus statique, en réservant des ressources avant la soumission du

travail. C'est le cas de JES3, décrit plus haut, avec une préallocation (device fencing) de périphériques dans le cas d'une planification fait intervenir des dépendances entre travaux (DJC).

Si l'allocation dynamique repousse le moment de l'allocation effective, l'allocation pré-emptive est associée à la possibilité de reprendre autoritairement une ressource préalablement allouée pour l'attribuer à un autre processus. Le processeur central est généralement géré de façon pré-emptive.

	PRE-EMPTIF	NON PRE-EMPTIF
STATIQUE	MC	imprimante
DYNAMIQUE	PC	espace disque

Figure 4g: exemples de types d'allocation.

STATIQUE :alloué en début d'étape pour une durée infinie
DYNAMIQUE :alloué en cours d'étape pour une durée infinie
PRE-EMPTIF:alloué en cours d'étape pour une durée limitée

4.3.3 – Problèmes d'interblocage.

L'allocation statique a le grand avantage d'être simple, donc de conduire à des SE plus simples, et surtout d'éviter un problème complexe, le problème des **interblocages** ou "étreinte fatale" (deadlock, deadly embrace). Cette situation est susceptible de se produire lorsque le SE fait de l'allocation **dynamique**: il permet à l'étape de commencer son exécution avant d'avoir acquis toutes les ressources nécessaires. On devrait plutôt écrire "avant qu'elle ait acquis toutes les ressources nécessaires autres que le processeur" puisque l'exécution correspond à l'attribution de la ressource processeur.

A s'exécute ⟶ [A obtient AX] ⟶ A demande BY

B s'exécute ⟶ [B obtient BY] ⟶ B demande AX

Figure 4h: interblocage entre les processus A et B.

Prenons l'exemple de deux étapes A et B de deux travaux différents: l'étape A s'exécute après avoir obtenu la ressource AX, l'étape B s'exécute après avoir obtenu la ressource BY (cf. figure 4h). En cours d'exécution, A demande la ressource BY; A sera donc mise en attente jusqu'à ce que B se termine et rende au SE la ressource BY. Notons que pendant tout ce temps, le SE doit gérer une étape inactive et donc perdre du temps (overhead, déperdition). Malheureusement B a besoin de la ressource AX pour se terminer. Le système est alors bloqué puisque A a besoin de BY possédé par B, et B attend AX que détient A. On trouvera dans [CRO75] et [PET83] une présentation plus complète et plus formelle de ce problème classique d'interblocage. [IBM77] indique que ceci est résolu sur MVS en attribuant sur un même processeur les verrous (lock) dans l'ordre croissant des priorités.

4.3.4 – Désallocation.

En fin d'étape le SE récupère les ressources attribuées afin de les donner à d'autres travaux ou aux étapes suivantes du même travail. Afin de réduire la charge des allocateurs, et/ou de ne pas ralentir le travail considéré, une étape peut indiquer que telle ressource ne doit pas être rendue au SE mais conservée pour la suite du travail.

C'est le rôle de l'option PASS dans le paramètre DISP des commandes // DD des OS d'IBM. Sous GCOS3 c'est le "code disposition" qui joue ce rôle: $ TAPE IN,A1S permet de "passer" le dérouleur A1 aux étapes suivantes, $ FILE S*,C2R rend l'espace occupé par le fichier S* en fin d'étape. Sous GCOS7, les paramètres END et ABEND de la commande $ ASSIGN permettent d'indiquer des dispositions différentes selon que l'étape se termine bien (END) ou mal (ABEND).

Afin d'améliorer la simultanéité, donc le débit global, la restitution des ressources au SE peut être faite de façon anticipée, dynamique donc, avant la fin de l'étape. Un fichier pourra être ainsi restitué au moment de sa fermeture (close). C'est le but du paramètre FREE (égal à END plus CLOSE) dans les commandes // DD SYSOUT de MVS. Les ressources sont normalement libérées lorsque l'initiateur signale la fin de l'étape.

Lorsque l'étape est terminée, tout ou partie des processus de l'étape est alors détruit, des informations comptables sont écrites sur le fichier comptable:

* heure de fin,
* temps PC utilisé,
* temps de résidence en MC (pour évaluer le va-et-vient),
* temps canal parfois,
* temps d'E/S et/ou nombre d'E/S sur les différents périphériques, avec le nombre d'erreurs,
* nombre de cartes lues,
* nombre de lignes imprimées,
* taille MC utilisée,
* taille disque utilisée, etc.

S'il y a d'autres étapes dans le travail correspondant, l'étape qui suit entre dans sa phase d'allocation. S'il n'y a pas d'autres étapes, le travail disparaîtra du système après que ses résultats accumulés sur le fichier de sortie du système (SYSOUT pour SYStem OUTput) auront été imprimés par le SE. Dans le cas de MVS, l'espace d'adresses sera attribué à un autre travail. L'étape peut se terminer:

- **normalement** si aucune erreur bloquante n'est apparue au cours de son exécution,
- **anormalement** (Abnormal Termination) en cas d'erreur :
  * due à l'étape elle-même
    . erreur de programme
    . dépassement des consommations autorisées ou annoncées
  * due à l'environnement
    . erreur matérielle impossible à surmonter,
    . demande de l'utilisateur ou du pupitreur.

Selon le système considéré, il sera possible de relancer (restart) le travail de façon plus ou moins automatique. En cas de fin anormale d'une étape, GCOS3 donne la main à une séquence de code de l'étape appelée "wrap-up", séquence que l'utilisateur peut programmer à sa convenance. La commande $ BREAK permet de définir un point de reprise dans le travail en cas de fin anormale d'une des étapes précédentes.

Les systèmes offrent généralement un moyen de communication entre les différentes étapes d'un même travail (en dehors des fichiers bien entendu) :

- code condition (Condition Code, CC) sur IBM/370,
- mot de commutation (Switch Word) sous GCOS3,
- variables système avec les langages de commande procéduraux (CPL sur Primos, DCL sur VAX/VMS, etc.).

Chaque étape, ainsi que le langage de commande (Control Language), peut donc tester la valeur de cette variable et en tirer la conclusion voulue.

La désallocation des ressources est réalisée par les "Terminator" sous OS d'IBM.

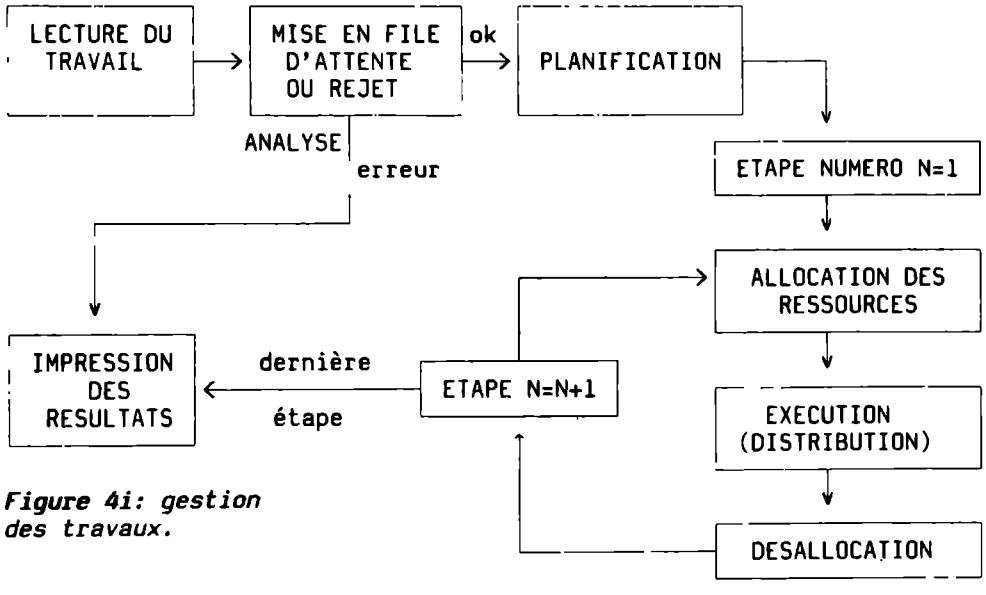


Figure 4i: gestion des travaux.

4.3.5 – Types, classes de ressources.

Le SE réalise une banalisation des ressources qui prend plusieurs aspects. Toutes les ressources d'un même type formeront une seule ressource globale. Ainsi l'espace disponible sur l'ensemble des disques formera une seule ressource référencée par la commande $ FILE sur GCOS3.

Pour les unités périphériques le SE fera apparaître des appareils **virtuels:**

- une unité **dédiée** est une unité utilisée de façon non partageable: elle est attribuée à une seule étape à la fois, par volonté délibérée ou par obligation liée à la nature physique du périphérique. Une imprimante par exemple ne peut pas être attribuée simultanément à plusieurs étapes, ceci conduirait à des pages contenant des lignes sans aucun rapport entre elles (le SE universitaire THE dont s'inspire par ailleurs Guardian, l'alloue cependant par page);
- une unité **partagée** est simultanément utilisable par plusieurs étapes. Deux étapes peuvent écrire sur deux fichiers différents, c'est-à-dire sur deux portions différentes du disque. Elles peuvent aussi lire le même fichier. Deux sessions peuvent passer (multiplexage) sur la même ligne de communication;
- une unité **virtuelle** fait appel à une technique qui permet d'utiliser de façon partageable un appareil qui est non partageable par nature et devrait donc être utilisé de façon dédiée. Ceci est fréquemment utilisé pour les périphériques lents comme les imprimantes ou les lecteurs de cartes.

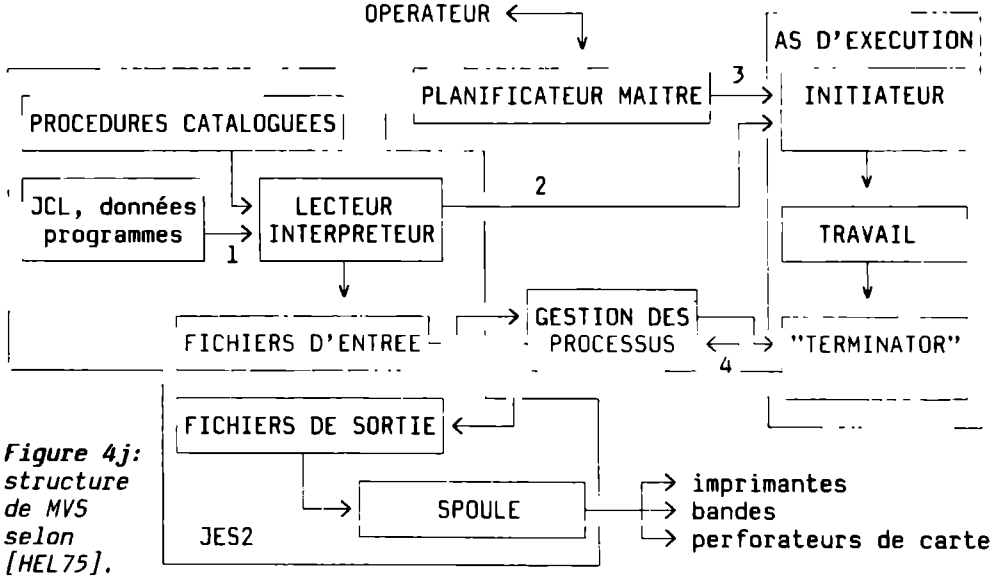


Figure 4j: structure de MVS selon [HEL75].

4.4 – Sous-système de sortie.

La dernière étape terminée, toutes les ressources sont rendues au système d'exploitation, et les résultats du travail sont imprimés. C'est le rôle du sous-système de sortie que de collecter toutes les sorties de toutes les étapes, de les organiser en fonction de différents critères, puis de les imprimer. On parlera de gestion des SYSOUT, le pendant du SYSIN vu dans le sous-système d'entrée. Comme ces sorties prennent beaucoup de place sur disque, et comme leur impression doit attendre la fin du travail, il est possible dans certains systèmes d'imprimer des fichiers de sortie avant la fin du

travail: fichier "backdoor" de GCOS3, fichier "spin-off" de JES. GOT dans GCOS3, JES dans le cas de MVS contiennent les fonctions des sous-systèmes de sortie des travaux.

Tout comme les requêtes sont réparties en classes dans certains SE, le mécanisme de classes est aussi utilisé dans le sous-système de sortie ("output class" de MVS avec le paramètre SYSOUT= dans les commandes). Il permet de regrouper les sorties analogues.

En fin d'étape sous MVS, l'initiateur est rappelé, désalloue les ressources, puis reprend le cycle d'allocation pour l'étape suivante.

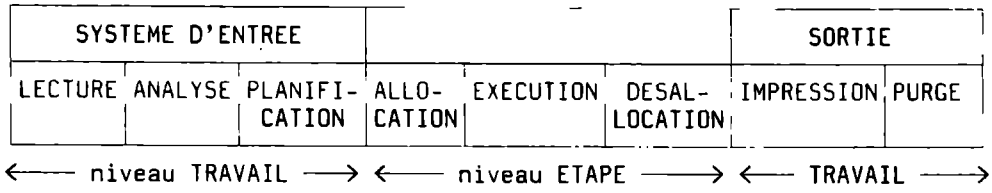


Figure 4k: structure du système d'exploitation.

4.5 – Gestion des travaux, enchaînement, noyau.

L'ensemble des composantes du système d'exploitation qui ont la charge de la lecture, l'analyse, la planification, l'impression et la purge des travaux constitue ce que l'on appelle le système (ou sous-système) de gestion des travaux (job management, cf. figure 4u), appelé aussi **moniteur d'enchaînement** par le passé.

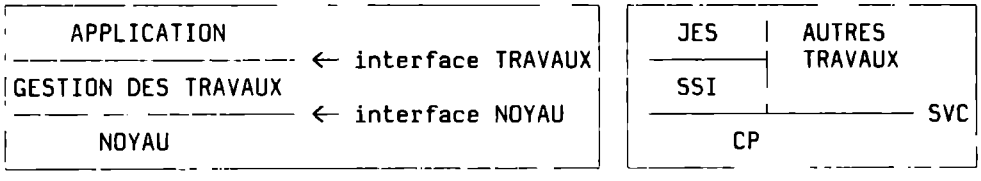


Figure 41: structure du SE faisant apparaître le noyau. *Cas de MVS*

Ces composantes sont à la périphérie du système d'exploitation dans de nombreux cas: elles utilisent un sous-ensemble du SE que nous appellerons le **noyau** (nucleus de MVS, exécutive, kernel, superviseur résident, etc.) comme le montre la figure 41. IBM utilise fréquemment le terme de CP (Control Program), SCP (System CP), BCP (Basic CP) pour parler du noyau (cf. [IBM80C], figure 4m).

Ainsi sous MVS, JES est lui-même un travail qui est automatiquement lancé après l'initialisation du noyau par le "planificateur maître" (master scheduler); JES utilise une interface de MVS créée pour les sous-systèmes, SSI comme "SubSystem Interface". Sous GCOS3, GIN, GOT, CALC, PALC, SCHED sont eux aussi vus comme des travaux par le noyau de GCOS3 (HCM).

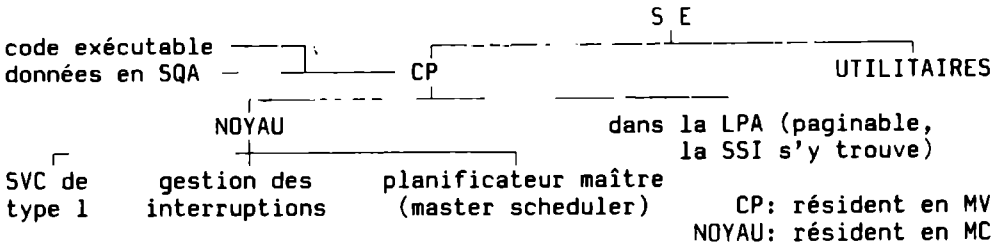
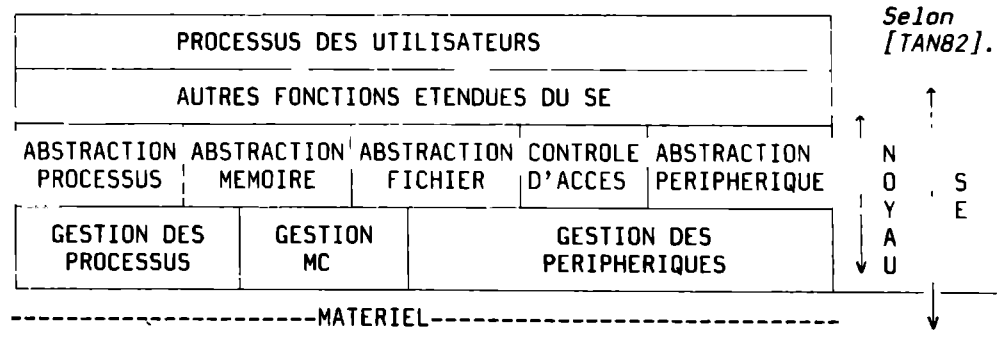


Figure 4m: place de CP dans les OS d'IBM.

Dans cet ordre d'idées, [MUL82A] décompose le logiciel système en deux parties, le "hardcore supervisor" et le "on-line software". Le "hardcore supervisor" est la partie du logiciel système qui ne peut être changée quand le système fonctionne. Le terme de noyau recouvre trois idées principales:

- une idée de **minimalité** des fonctions implémentées. Le noyau est aussi petit que possible, il peut donc faire l'objet d'une maintenance moins onéreuse, être plus **sûr** dans son fonctionnement;
- une idée de **résidence** permanente en mémoire centrale, résidence facilité par sa petite taille, fonctionnelle et donc physique;
- une idée de **mode privilégié** car le code du noyau se déroule généralement dans ce mode de fonctionnement du processeur.

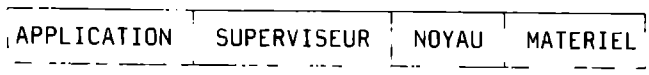
Une quatrième idée s'est renforcée avec l'évolution, celle de la structure en **couches** du logiciel. Au dessus d'un noyau offrant un minimum de fonctionnalités, on écrit une seconde couche offrant un service un peu plus riche, et ainsi de suite. L'utilisateur pourra ainsi se brancher à différents niveaux, selon ses contraintes de performance, de portabilité et selon ses compétences techniques.



Selon [TAN82].

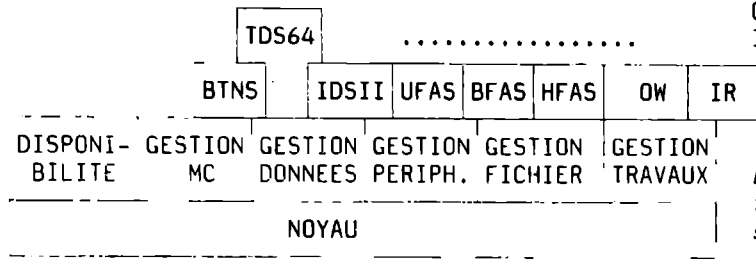
Les idées de couche et surtout de minimalité prennent une importance accrue dans les noyaux "sécurisés" des systèmes qui ont comme objectif d'offrir un niveau élevé de protection, sécurité, confidentialité (cf. chapitre 14). [AME83] distingue les notions de noyau et de superviseur par la minimalité du noyau, le fait qu'il gère les ressources partagées entre plusieurs utilisateurs, qu'il est donc res-

ponsable de la protection et de la sécurité:



Les couches selon [AME83].

Le terme de noyau peut aussi être limité à un ensemble très restreint de fonctionnalités, à celles fournies par l'interface matériel-logiciel, microprogrammée dans le cas de [BUL77], cas rappelé figure 4n.



OW: output writer
IR: input reader

Figure 4n: structure de GCOS64 selon [BUL77].

Il ne faut pas prendre à la lettre les représentations générales en couches qui font apparaître les applications comme coupées de l'accès au matériel par le SE. Dans la figure 4o, le schéma de gauche signifie seulement que les applications utilisent les services du SE qui lui s'appuie sur le matériel. Il est bien évident que lorsque l'application s'exécute (a le contrôle du processeur), elle a directement accès au matériel, ce que montre le schéma de droite.

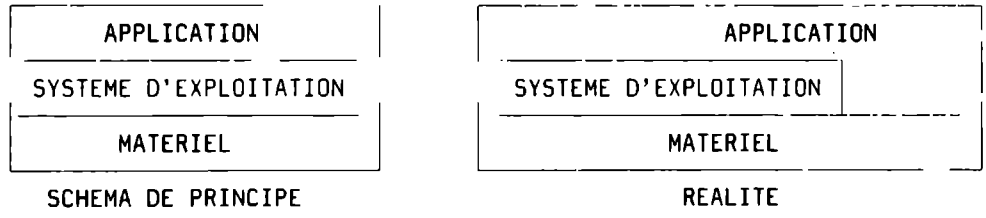


Figure 4o: portée des représentations en couches.

4.6 – Machines multi-SE.

Un SE est conçu pour être le **seul** décideur au sein du système informatique qui apparaît ainsi sous une nature **centralisée**. Lui seul a une connaissance de l'état de **toutes** les ressources. Pour un certain nombre de raisons il existe des systèmes disposant de plusieurs SE, identiques ou non, fonctionnant simultanément ou non, avec ou sans relations maître/esclave, que l'on trouve dans plusieurs contextes:

- SE coiffés par un embryon de SE de plus haut niveau de responsabilité, spécialisé dans cette tâche de gestion des SE et généralement appelé **hyperviseur** (cf. chapitre 10). L'hyperviseur a alors pour rôle de recréer l'autorité **centrale** et de distribuer les ressources aux SE qui les allouent à leur tour aux requêtes qu'ils

gèrent. Chaque SE peut être différent;

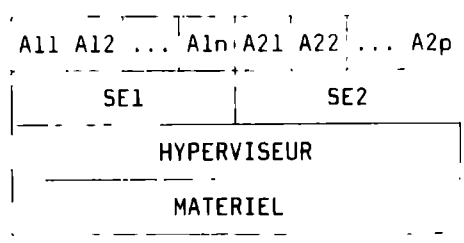


Figure 4p: système multi-SE avec hyperviseur.

Aij: application sous SEi

- SE coiffé par un autre SE qui joue en même temps le rôle ci-dessus de l'hyperviseur. Cas du mode "accommodation" (ou "encapsulé") de Multics qui permet à GCOS3 de s'exécuter sur le même matériel que Multics. Cas aussi de GCOS64 et GCOS7 avec les SE associés aux "décors";

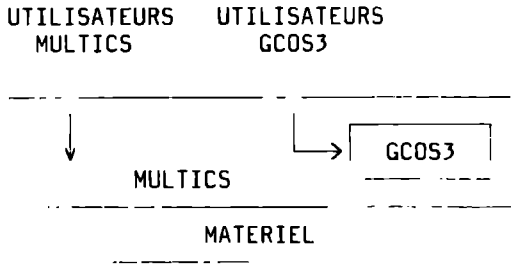
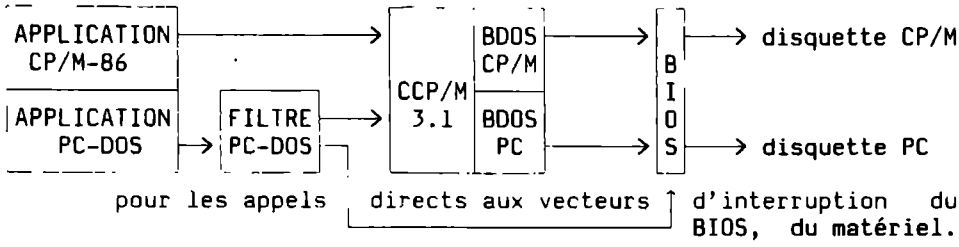


Figure 4q: cas d'un SE "accommodé" par un autre SE.

- SE **distribués** dans lesquels nous rangerons:
  * les **SER** ou "SE Réseau", c'est-à-dire des extensions des fonctionnalités des SE qui permettent le dialogue entre utilisateurs et applications sur des systèmes interconnectés par des réseaux, locaux ou non (cf. chapitre 13). Chacune des copies d'un SER ne gère qu'un sous-ensemble de la ressource globale constituée par tous les éléments du réseau. Tous les SE ne sont pas obligatoirement identiques: cas des différents GCOS sous DSA, de MVS, DOS et autres sous SNA, de MS-DOS et CP/M avec Netware. PCNet d'Orchid par contre ne fonctionne qu'avec MS-DOS: toutes les copies des SE doivent être identiques;
  * les **SED** (SE dupliqués), proches des précédents, sinon cas particulier, conçus pour des multisystèmes dans lesquels chaque processeur dispose d'une copie **identique** du SED, tout en donnant la visibilité d'un seul système à ses utilisateurs. Cas de Guardian sur Tandem;
  * les SE de multimicros où un noyau très restreint est dupliqué sur chacun des micros, les autres fonctions étant distribuées sur les différents micros;
- systèmes dans lesquels il y a une **partition** des ressources, comme dans le cas des SED, mais où la coopération entre les SE n'est pas un des éléments de base de la conception du système. Des exemples en sont:

* Goupil3 de SMT qui fonctionne normalement sous FLEX-9 peut être doté d'une carte basée sur un Z-80 afin de le faire fonctionner sous CP/M. La partition est alors très prononcée puisqu'un seul des deux SE peut fonctionner à un moment donné, et il n'y a pas, par exemple, possibilité d'accéder aux fichiers créés sous un SE lorsque l'autre tourne. Au début des micro-ordinateurs commercialisés par les grands constructeurs, de nombreux cas semblables se présentèrent pour ceux qui avaient "raté" le train CP/M, et surtout MS-DOS;



* le PC d'IBM peut être équipé d'une carte P-Sceptus ([SAL86]) basée sur un 68008 de Motorola dans laquelle tourne un SE temps réel inspiré du noyau Sceptre du BNI, donnant ainsi la dimension "temps réel" au PC. Dans cet exemple les deux SE peuvent tourner simultanément, MS-DOS constituant un environnement de développement et de chargement pour le SE temps réel;
* l'option MDF (Multiple Domain Facility, matériel et microcode) annoncée en 1984 par Amdahl permet à deux SE de tourner sur un même processeur, dans des domaines logiques séparés; la séparation est réalisée par du microcode, les deux SE ayant chacun leur machine. [MOA86] estime que lorsque MVS/XA et VM/SP HPO y tournent simultanément, la dégradation par rapport au fonctionnement monoposte est de 5 % seulement;

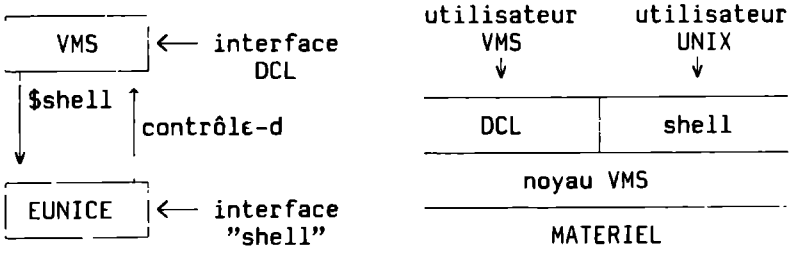


Figure 4r: SE avec plusieurs interfaces JCL.

- SE dont les fonctionnalités ont été "plaquées" (mapped) sur celles d'un autre SE qui est en fait le seul à tourner dans l'ordinateur. C'est le cas de nombreuses implémentations d'Unix . Ainsi EUNICE sous VMS offre à un utilisateur l'utilisation du langage de commande d'Unix (le Shell) sur une machine gérée par VMS (figure 4r). Dans le cas d'Eunice les objets du système sont visibles par les deux envi-

ronnements : un fichier créé par un programme sous Eunice sera manipulable sous VMS. Il y a en fait remplacement du langage de commande natif de VMS (DCL comme DEC Control Language) par le "Shell" d'Unix. Il en est de même dans Primix, obtenu en ajoutant au noyau de Primos 19.4.2 les fonctions nécessaires pour offrir les fonctionnalités d'Unix, dans VUE (VOS Unix Environment) de Harris où le Shell d'Unix est mis au-dessus du noyau de VOS.

4.7 – SE multiprocesseur/système.

A l'heure actuelle il y a une grande poussée des machines comportant plusieurs processeurs, quelques processeurs de façon classique, plusieurs centaines, voire plusieurs milliers au niveau de la recherche et des laboratoires. Nous en avons parlé dans le tome I dans les chapitres sur le parallélisme et l'anticipation. Certains SE sont conçus pour des monoprocesseurs (Unix, MS-DOS), d'autres pour des multiprocesseurs (GCOS3, OS1100), d'autres encore pour des multisystèmes (Guardian, Cortex, ACK). Unix a été modifié dans les deux directions par différents constructeurs:

- vers des multiprocesseurs: cas de Sequent Computer Systems,
- vers des multisystèmes: cas de Flexible Computer Corp. qui commercialise une machine FLEX/32 avec 20 processeurs sous Unix.

Mentionnons encore le cas de Kronos de CDC qui tourne en majorité dans les processeurs périphériques (PP, cf. tome I), quelques fonctions pouvant être exécutées dans les PC à la demande des PP.

oOo

Les figures 4t à 4u donnent quelques exemples de la façon dont les constructeurs présentent la structure des composantes de leur logiciel.

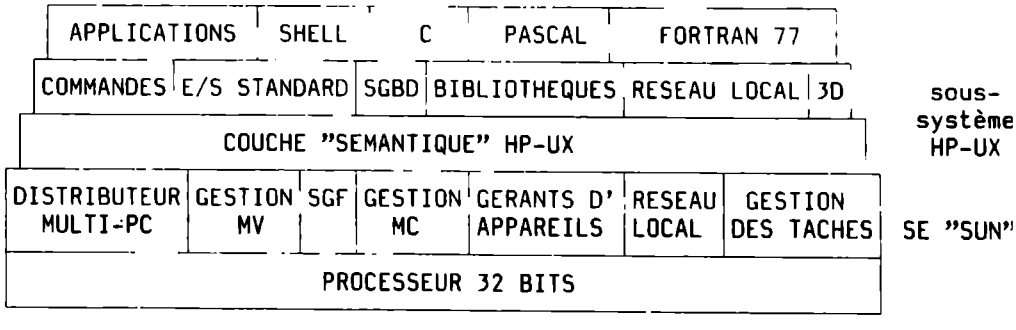


Figure 4s: HP-UX de Hewlett-Packard.

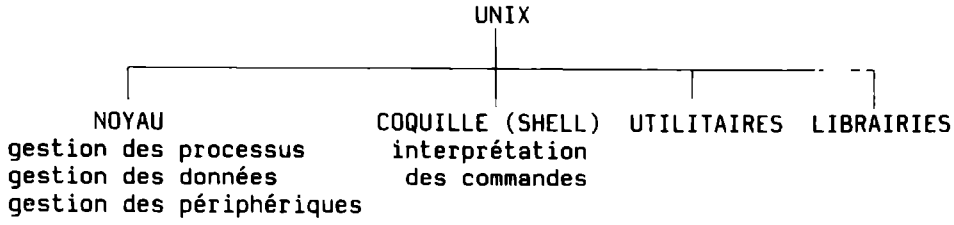


Figure 4t: les composantes d'Unix.

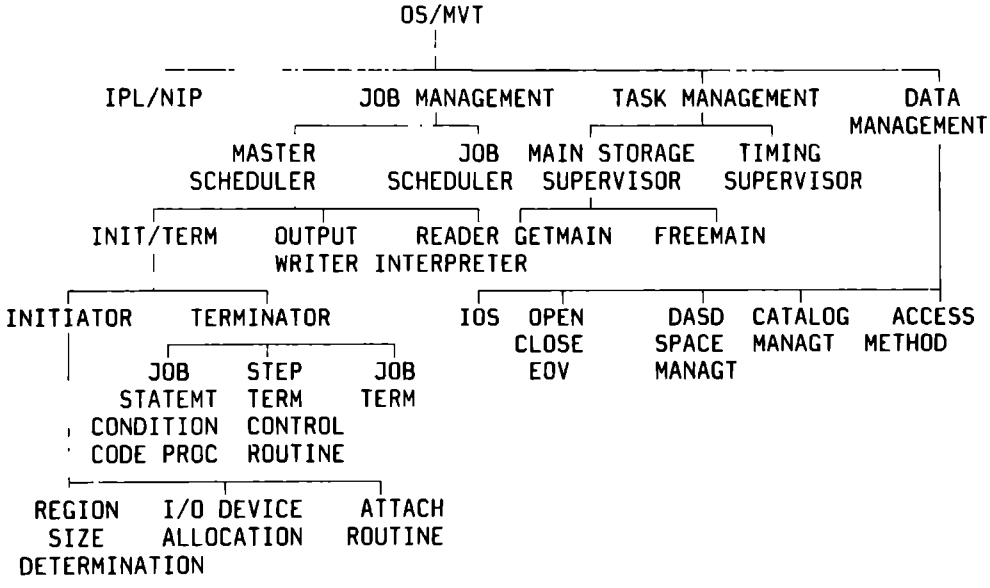


Figure 4u: les composantes de MVT selon IBM.

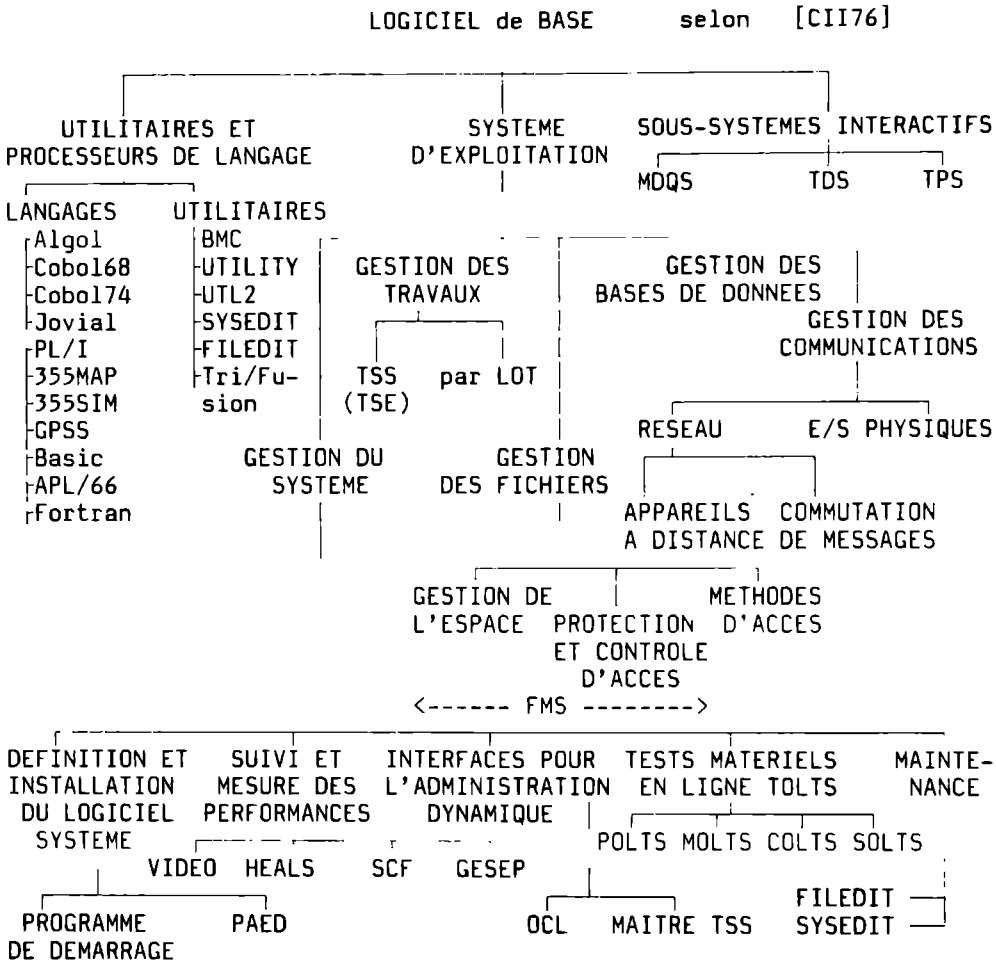


Figure 4v: les composantes de GCOS3 selon Bull.

chapitre 5

allocation des ressources

5.1 – Notion de processus.

Nous avons vu dans le chapitre précédent où se situait l'allocation des ressources dans le cheminement des requêtes à l'intérieur du système d'exploitation:
- au début de chaque étape dans un SE orienté vers le mode différé,
- lors de la connexion (login) dans un SE orienté vers le mode dialogué, et lors de chaque commande,
- en cours d'étape ou de session s'il y a allocation dynamique.

L'étape est une notion intermédiaire dans un système d'exploitation: elle correspond à un découpage statique du travail en éléments de travail exécutés les uns après les autres. Le travail une fois planifié et les ressources périphériques allouées, le SE verra plutôt chaque étape comme un ou plusieurs **processus** (process), et certaines ressources, comme le processeur central, seront attribuées à un processus.

Le processus peut être défini comme le plus petit ensemble de logiciel auquel on peut attribuer des ressources. Le SE ne connaît rien en dessous du niveau du processus, comme le résume la figure 5a.

lot	: unité de soumission
travail	: unité de planification
étape	: unité d'allocation
processus	: unité d'exécution

Le processus est encore un ensemble formé:
- d'un code exécutable provenant du programme source du programmeur augmenté des librairies introduites lors de l'édition de liens,
- de données,
- d'un contexte définissant toutes les ressources qu'il possède: une image des registres, du mot d'état du processeur (dont le compteur ordinal), des blocs de contrôle décrivant les fichiers qui lui sont attribués, etc.

Contrairement au travail qui est une construction externe au noyau, le processus est une construction de ce noyau, et dans certains cas, directement supporté par le matériel (DPS7 de Bull, iAPX286, VAX par

exemple). Le matériel du DPS7 implémente d'ailleurs une notion proche de celle de l'étape ou du groupe de processus puisque ce qui est appelé travail (job) au niveau du matériel est en fait un ensemble de 1 à 256 processus. VMS définit ([VAX32]) un "job" comme «une famille de processus partageant les mêmes ressources. Il est l'entité de comptabilisation de VMS». De même iRMX86 définit un travail comme un groupe de processus (cf. [FON8X]).

Selon les SE on parlera aussi de **tâche** (task) pour processus, de "programme concurrent" ([WIT84]). Le terme de "processeur virtuel" est parfois utilisé; il correspond au fait qu'un processus contient une image de toutes les ressources du processeur: lorsqu'il perd le contrôle du processeur on copie dans une de ses structures de contrôle (PCB comme "Process Control Block", ou TCB comme "Task Control Block", TSS dans l'iAPX286) le contenu du mot d'état du processeur (PSW), le contenu des registres. Lorsque le processus a de nouveau la main (il reprend le contrôle du processeur) les registres du processeur sont initialisés avec la photo prise dans le PCB. L'expression de "SE concurrent" (concurrent system) est liée à l'idée de compétition des processus entre eux pour obtenir le processeur central et les autres ressources. L'usage des termes étant peu rigoureux en informatique, on trouvera dans AOS/VS de Data General qu'il peut y avoir jusqu'à 255 processus simultanés, chacun comprenant jusqu'à 32 tâches.

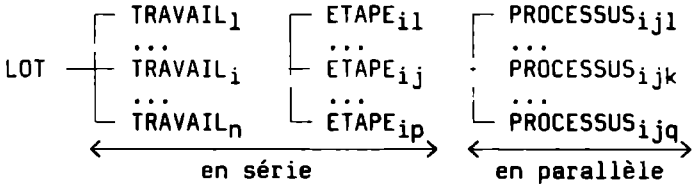


Figure 5a: relations entre les formes des requêtes.

5.1.1 – Création des processus.

La première étape d'un travail donne lieu à la création d'un ou plusieurs processus. L'application écrite par l'utilisateur provoque la création d'un processus au minimum: c'est le cas des programmes classiques de gestion, des compilations. Certains systèmes d'exploitation vont y ajouter d'autres processus: MVS ajoutera ainsi un processus appelé RCT (Region Control Task) qui jouera un rôle de processus maître pour toute la vie de l'espace d'adresses (AS) et qui subsistera dans tous les travaux (non TSO) exécutés dans cet AS; à ce processus RCT seront chaînés d'autres processus (DUMP, l'initiateur, puis le processus défini dans la commande // EXEC).

A la fin de chaque étape il pourra y avoir destruction des processus (ou seulement de certains d'entre eux), ou bien conservation pour l'étape suivante. Dans les systèmes orientés vers le mode dialogué, chaque commande ne conduit généralement pas à la création d'un nouveau processus, mais seulement au changement de la forme exécutable (cf. tome V, primitive EXEC d'Unix) déroulée par le processus. La création d'un processus étant une opération relativement lourde, on l'évite dans la mesure du possible.

Selon les SE, le processus sera créé plus ou moins tôt: avant allocation des ressources périphériques, ou après attribution de ces ressources périphériques. Création veut dire inscription du processus dans les structures de contrôle du noyau.

Dans certains SE cette structure est une table à n entrées, n étant le nombre maximum de processus supportés par le SE. C'est le cas de GCOS3 qui supportait par exemple 64 processus au plus à la fin des années 1970; nous parlons ici des processus connus de GCOS3: nous verrons que d'autres n'en sont pas directement connus. Dans d'autres SE, les structures de contrôle des processus, que nous appellerons PCB, sont chaînées entre elles. Ce nombre maximum de processus simultanément actifs correspond dans de nombreux SE à l'expression "supportant jusqu'à n utilisateurs" que l'on rencontre dans les SE interactifs.

Un des avantages retirés d'une structure multiprocessus est la meilleure compétitivité de l'étape pour obtenir la ressource processeur puisqu'elle a plusieurs candidats. Le coût à payer est un accroissement de la consommation improductive du processeur (déperdition, overhead) provoquée par la commutation des tâches réalisée par le SE.

La macro ATTACH de MVS permet de créer une une tâche de priorité inférieure ou égale à celle du créateur. Sous Multics le processus "initializer" vu précédemment crée et détruit les autres processus. La primitive FORK d'Unix permet de créer un nouveau processus qui possède aussi les fichiers de son créateur, d'où la mise en place facilitée des tubes (pipe).

5.1.2 – Etats d'un processus.

Depuis sa création jusqu'à sa destruction, chaque processus va passer par un certain nombre d'états:

- en attente (waiting) de ressources ou d'événements. On distingue en fait au moins deux états:
  * l'état "en attente" d'événement comme une fin d'entrée/sortie, ou la satisfaction d'une demande de ressource;
  * l'état **bloqué** (hold) ou suspendu dans lequel le processus attend une intervention humaine, du pupitreur par exemple, pour passer dans l'état en attente;

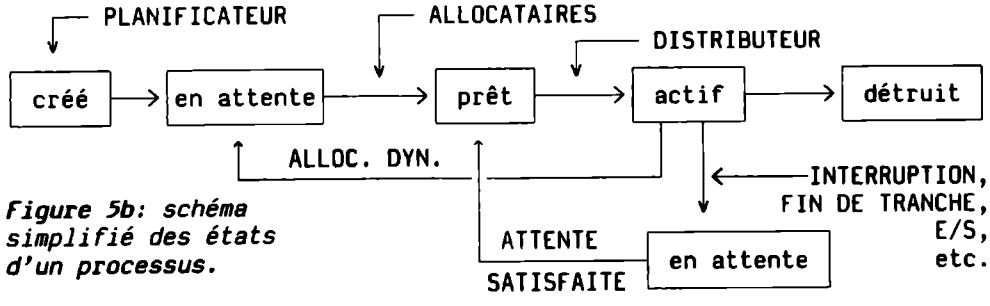


Figure 5b: schéma simplifié des états d'un processus.

- **prêt** (ready) lorsque le processus dispose de toutes les ressources nécessaires autres que le processeur central. On parlera aussi de processus "éligible" (eligible);
- **actif** ou "en cours d'exécution" (active, running) lorsque le processeur central lui a été attribué et qu'il déroule donc ses instructions. Il y a un seul processus actif par processeur central.

Une fois l'étape terminée, le processus est détruit. Le nombre d'états de la figure 5b augmente si l'on fait intervenir les mécanismes de vidage (swapping) de certains SE, mécanismes permettant de récupérer la portion de mémoire centrale occupée par une étape en la mettant temporairement sur mémoire auxiliaire (disque typiquement). Seuls les processus en mémoire ("swapped in") sont susceptibles d'être actifs.

5.1.3 – Relations entre processus.

Les processus des différentes étapes en cours d'exécution n'ont a priori aucun rapport entre eux et chacun s'exécute en ignorant totalement les autres processus des autres étapes.

Lorsque le SE le permet et que l'application le nécessite, une étape comporte plusieurs processus. Les différents processus de la même étape peuvent alors:

- se dérouler indépendamment les uns des autres en dehors des communications et synchronisations **explicites** qu'y a mises le programmeur;
- dépendre les uns des autres. Une réalisation classique de cette dépendance est la **filiation** entre les processus: le processus qui en crée un autre sera appelé processus "père", le processus créé sera dit "fils", et le père aura des droits sur ses fils. Tout comme il a pu les créer, le père peut les supprimer. Ces processus partagent généralement les mêmes ressources, ce qui est logique quand ils appartiennent à la même étape. On trouve ceci sous Unix pour les fichiers. Primos en version 18, Multics, n'ont pas cette filiation de processus, bien que dans une même session l'utilisateur puisse lancer plusieurs processus en parallèle ("phantoms" de Primos) et/ou lancer des travaux "batch". Sous VMS le "père" sera appelé "processus propriétaire" de ses sous-processus (subprocess) et son arrêt entraîne celui de ses sous-processus; seuls les processus dits "détachés" (detached) n'ont aucun propriétaire, c'est le cas du processus qui dialogue avec un utilisateur connecté.

Le SE peut lui-même être constitué sous forme de processus ou de groupes de processus (cas de GCOS7).

5.2 – Allocation d'espace sur disque.

L'allocation d'espace sur disque est l'une des allocations de ressources périphériques. Nous verrons successivement la façon d'allouer l'espace aux fichiers, la gestion de cet espace sur le disque, celle de l'espace d'un fichier, puis nous présenterons la gestion des portions défectueuses du disque, le choix du disque sur lequel est créé un fichier, pour terminer sur des considérations de sécurité et de confidentialité.

5.2.1 – Allocations statique et dynamique.

Les systèmes d'exploitation classiques orientés vers le mode différé procédaient (procèdent encore pour certains) par allocation **statique**: lorsqu'un fichier est utilisé par une étape, il doit être alloué en début d'étape, avec la **taille maximum** qu'il atteindra un jour dans le cas d'un nouveau fichier. Il atteindra cette taille maximum au cours de l'exécution de l'étape qui l'a créé, ou bien au cours de l'exécution d'une autre étape du même travail, ou même d'un autre travail, dans quelques minutes ou dans quelques mois. Il pourra d'ailleurs ne jamais atteindre cette taille maximum tant il est difficile de préjuger de la taille maximale d'un fichier.

Dans certains SE, les étapes ont la possibilité de rendre l'espace inutilisé lorsque le fichier est fermé (par l'instruction CLOSE de Cobol par exemple). L'option RLSE dans le paramètre SPACE de la commande // DD des OS d'IBM joue ce rôle.

Cette allocation statique conduit donc à une **mauvaise utilisation** du support disque puisqu'une partie de la place ne sera jamais utilisée. Elle est par contre plus **performante** car elle permet d'attribuer une seule zone contiguë sur le disque, ce qui diminuera les mouvements de bras (cf. tome IV) et accélérera les entrées-sorties, tout en minimisant la taille des structures nécessaires à sa gestion. Cet argument de performance est totalement valable lorsqu'un seul fichier est actif sur le disque, situation favorisée par la monoprogrammation. Avec les SE offrant la multiprogrammation (exécution de plusieurs travaux en parallèle), cet avantage s'estompe, d'autant plus lorsque la taille des transferts est peu importante.

Le système des extensions permet d'introduire un premier aspect **dynamique** au niveau de l'allocation: la taille maximum ne sera pas allouée automatiquement lors de la création du fichier, seule une première allocation partielle d'espace sera faite, et lorsque cet espace "primaire" sera rempli il y aura allocation d'une seconde portion d'espace (une extension, extent), puis d'une troisième si nécessaire, etc. On voit ainsi apparaître une allocation **dynamique** au niveau des espaces secondaires.

Les quanta permettent une allocation dynamique générale et banalisée. Lors de la création du fichier on lui alloue un quantum. Quand ce quantum est plein, un second quantum lui est attribué, etc., jusqu'à atteindre une taille limite que le fichier ne doit pas dépasser, taille limite qui n'existe pas dans tous les SE (cas de Primos), ou n'est pas systématiquement activé (VMS entre autres). Ce quantum est souvent appelé "cluster" (cas de VMS, MS-DOS).

Cette technique d'allocation est utilisée sur la grande majorité des SE orientés vers le mode dialogué: le SE n'y étant pas maître de la charge (workload) que soumettent les utilisateurs, il serait illusoire d'essayer de regrouper les articles d'un même fichier, les différents utilisateurs provoquant le déplacement des têtes de lecture de façon incontrôlable pour le SE ("vol de bras"). Plus généralement ils sont moins orientés vers la production, c'est-à-dire le traitement de

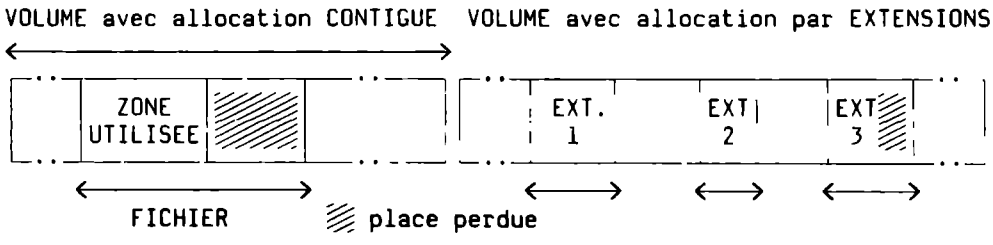
masse (voir chapitre 7 des exemples de taille de fichier sous Unix). La grande majorité des mini-ordinateurs relève aussi de cette catégorie. On le trouve encore dans nombre de SE de moyens et gros systèmes chez les concurrents d'IBM: cas de GCOS3.

5.2.2 – Allocation contiguë.

Lorsqu'il faut attribuer de l'espace pour un fichier sur un disque, se pose le problème de la position relative des zones attribuées. On dira que deux secteurs sont contigus s'ils se suivent sur la même piste, que deux pistes sont contiguës si elles se suivent dans le même cylindre.

La contiguïté présente un avantage mais aussi un inconvénient: il n'est pas évident de trouver une place contiguë suffisante pour chaque fichier. Ainsi dans le P-SYSTEM, un SE pour micro-ordinateurs compatibles avec le PC d'IBM, on a affaire à une allocation contiguë et si le disque ne contient pas un "trou" suffisant il sera impossible de créer le fichier. Il faudra alors "réorganiser" le disque en y tassant tous les fichiers afin de créer un trou maximum. Nous retrouverons des problèmes analogues dans la gestion de la mémoire centrale.

Dans certains SE où la contiguïté n'est pas obligatoire, elle peut être forcée: pour l'espace primaire (ou principal, primary, la première quantité allouée) sur les OS d'IBM avec l'option CONTIG dans le paramètre SPACE des commandes // DD. La commande FDL "FILE CONTIGUOUS" de RMS sous VMS force aussi l'allocation contiguë, alors que "FILE BEST_TRY_CONTIGUOUS" la sollicite si elle est possible.



Une évolution de cette allocation a conduit à créer pour un même fichier plusieurs zones contiguës (plusieurs extensions, extent). Il n'est plus nécessaire de trouver un "trou" suffisamment grand pour tout le fichier.

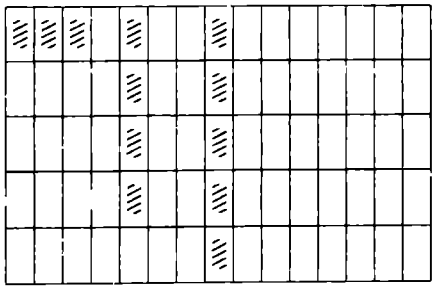
Les OS d'IBM offrent ce type d'allocation, et le paramètre SPACE=(CYL,(p,s)) permettra de créer un fichier dont l'espace primaire occupera p cylindres (option CYL), et chaque espace secondaire s cylindres. Le premier espace alloué, le primaire, peut contenir jusqu'à 5 extensions (4 s'il y a une étiquette, paramètre LABEL=SUL); un fichier peut avoir jusqu'à 16 extensions par volume, n fois 16 s'il est stocké sur n volumes (n disques) différents (cas de fichiers "multivolumes"). Sous DOS/VSE la commande // EXTENT permet de définir chaque extension d'un fichier. RMS (Record Management Services) de VMS utilise aussi cette notion d'extension, la taille de la première

extension étant définie par la commande FDL (File Definition Language) "FILE ALLOCATION"; pour la taille des autres extensions on utilise le paramètre RMS_EXTEND_SIZE de **VMS** qui définit la valeur de l'extension par défaut au niveau du système; les commandes DCL SET RMS_DEFAULT/EXTEND_QUANTITY ou SET FILE/EXTENSION, la commande FDL FILE EXTENSION définissent cette taille au niveau du processus, en nombre de "clusters", le "cluster" étant l'unité d'allocation de VMS, c'est-à-dire un nombre entier de "blocs"; le qualifieur /SYSTEM permet de modifier la valeur au niveau système dans les commandes DCL. Dans **RTE-A** toutes les extensions sont de même taille que l'espace primaire. **Guardian** sur Tandem utilise aussi le système des extensions, une principale (primary) et 15 secondaires qui sont la source du partitionnement d'un fichier sur plusieurs volumes ou sur plusieurs machines d'un réseau Expand.

En poussant jusqu'à l'extrême la méthode des extensions, on aboutit à l'allocation par **quantum** qui rompt avec toute contrainte de contiguïté: le fichier est découpé en portions de tailles identiques et tout fichier occupe un nombre entier de ces quanta, (un cluster contient un secteur sur les disquettes simple face, deux secteurs sur les double face). Le quantum peut être:

- fixe pour le système: un secteur sous FLEX9;
- fixe par type de disque: un secteur sur les disquettes simple face de MS-DOS, deux sur les double faces. Sur RTE-A il est d'un secteur de 256 octets sur les disques de 33,5 Mo, il passe à deux secteurs sur les disques de 67 Mo et à quatre secteurs sur ceux de 134 Mo;
- défini séparément pour chaque disque: cas de CP/M avec le paramètre BSH dans les DPB (Disk Parameter Block), cas de GCOS3 où l'unité d'allocation (AU comme Allocation Unit) est exprimée en "llinks" (little link) et rangée dans le treizième mot de l'étiquette du disque (disk pack label);
- défini séparément pour chaque fichier même.

Nous retrouverons cette technique de quantum dans la gestion des mémoires centrales avec la pagination.



Carte des "clusters" alloués à un fichier. Ce type de résultat peut être obtenu sous MS-DOS avec les utilitaires Norton ou PCtools. Chaque colonne représente une piste , chaque case les secteurs d'un même "cluster".

Notons qu'une copie logique sur une disquette vierge regroupera ceux d'un même fichier alors qu'une copie physique les recopiera tels quels.

Les exemples que nous avons cités pour le VAX illustrent le fait que les paramètres sont définis à des niveaux différents:

- lors de la génération du système: exemple de RMS_EXTEND_SIZE;

- lors de la création du fichier: cas des commandes FDL;
- au moment de l'exécution du programme ("run time"): cas de la commande DCL SET.

Une autre remarque nous conduit à souligner la différence entre type d'allocation et contiguïté: on peut avoir des quanta chaînés alloués de façon statique.

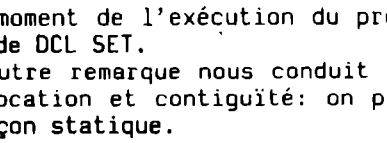


Figure 5c: types d'allocation d'espace sur disque et de contiguïté préférentielle.

5.2.3 – Gestion de la place sur disque.

Le SE doit être à même de savoir à tout instant quelle est la place disponible sur chaque disque, et où se situe cette place libre. Il utilise pour cela des méthodes qui relèvent de deux grands types:
- méthodes centralisées,
- méthodes décentralisées.

Dans les méthodes **centralisées**, le SE gère une table centrale qui liste toutes les zones disponibles sur le disque: c'est le cas de MS-DOS avec la FAT (File Allocation Table, table d'allocation des fichiers) qui contient, pour un volume (disquette ou disque) donné, une entrée par unité d'allocation (cluster). Chaque entrée indique si l'unité est libre ou non. Cette méthode est une extension de la technique "bit map" (carte de bits) dans laquelle chaque bit indique si le quantum associé est libre (valeur 1 par exemple) ou occupé (valeur 0).

GCOS3 utilise cette méthode avec une table du type "bit map" appelée AST (Available Space Table). Table qui peut être mise résidente en mémoire centrale pour améliorer les performances ("in core AST"), mais crée un risque de perte ou d'incohérence des données en cas de panne si le SE ne dispose pas de mécanismes de reprise (recovery) suffisants pour rétablir une situation saine. Dans Tripos (cf. [POU86]) le processus associé à chaque disque est chargé de réecrire cette "bit map" toutes les 3 secondes.

BLOC	CONTENU
0	étiquette de volume
1	table des "llinks" défectueux
2	table de reprise (recovery) du système dans ST1
3	table des pistes de remplacement
4	historique de l'espace défectueux
5-n	table de l'espace disponible (AST)
.	
n+1	pistes de remplacement
p	pistes pour les tests et diagnostics

Figure 5d: structure d'un disque sous GCOS3.

Dans les méthodes **décentralisées** il n'y a pas de table centrale,

chaque zone libre sur le volume contient l'adresse de la zone libre suivante et l'on a ainsi affaire à une liste de zones libres. Au départ le volume est initialisé comme une seule zone libre, ou bien comme une liste de quanta en cas d'allocation dynamique par quantum. FLEX9 relève de cette dernière catégorie, avec une chaîne (liste) des secteurs libres.

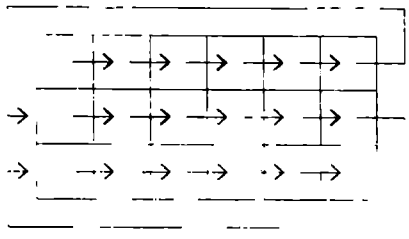


Figure 5e: exemple de volume vierge avec 18 unités d'allocation chaînées entre elles pour former une liste des unités disponibles.

5.2.4 – Organisation de l'espace d'un fichier.

L'espace attribué à un fichier est susceptible de contenir:
- les **données** écrites par l'utilisateur;
- des informations de contrôle ajoutées par les méthodes d'accès (voir chapitre 8) si nécessaire;
- des informations de contrôle ajoutées par le gestionnaire de l'espace disque si nécessaire;
- des zones allouées mais non encore écrites.

Nous ne nous intéressons ici qu'au troisième type de données et ignorons les informations technologiques (adresse du bloc ou du secteur, longueur des zones clé et données) présentées dans le tome IV et inscrites lors du "formattage" du volume.

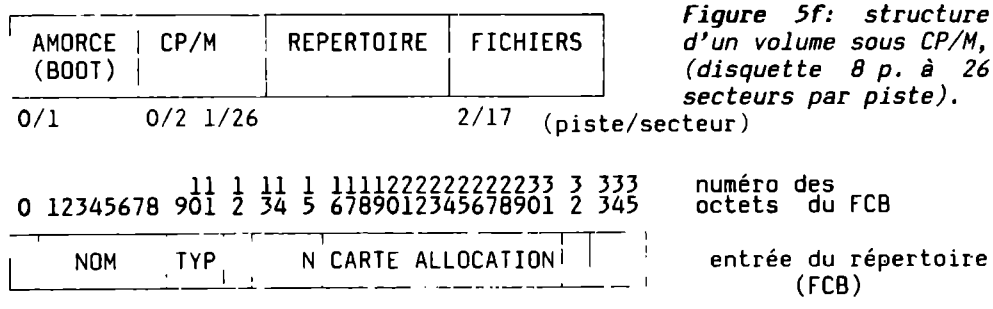


Figure 5f: structure d'un volume sous CP/M, (disquette 8 p. à 26 secteurs par piste).

Les informations de contrôle ajoutées par le gestionnaire de l'espace disque sont très réduites dans le cas d'une allocation contiguë. Dès qu'il n'y a plus allocation contiguë, il faut ajouter les éléments nécessaires pour localiser l'ensemble des données:
- de façon **centralisée** au moyen de tables contenant les adresses de chaque extension; cette méthode n'est valable que si le nombre d'extensions est limité, sinon il y a perte de place pour les petits fichiers. MS-DOS utilise une technique centralisée sans réservation

statique par fichier: ce sont les entrées de la FAT qui sont chaînées entre elles lorsqu'elles appartiennent à un même fichier, il y a en fait une réservation par unité allouable.

- une première **décentralisation** conduit à adapter la taille de cette table à la taille du fichier. La table est en fait un ensemble de sous-tables chaînées. C'est le cas de CP/M où un fichier peut représenter jusqu'à 16 extensions de 16 secteurs sur disquette; le répertoire (cf. figure 5f) contient donc de 1 à 16 entrées (un FCB pour "File Control Block", bloc de contrôle de fichier) par fichier. Chacun des 16 octets (16-31) de la figure 5f indique si le secteur associé est utilisé ou non, l'octet 15 (N) indiquant le nombre de secteurs utilisés. Un autre exemple est **Unix**: la structure générale d'un disque géré par Unix est schématisée figure 5h; la table des "i-nodes" contient une entrée par fichier stocké sur le volume. Chaque entrée contient les adresses des 10 premiers blocs alloués au fichier concerné. Si le fichier nécessite plus de 10 blocs, l'entrée contient l'adresse du bloc décrivant les 128 blocs suivants. On voit ainsi qu'un fichier ayant entre 11 et 138 blocs de données occupera entre 12 à 139 blocs puisqu'il faut un bloc d'adresses pour adresser les 128 derniers blocs. L'adressage de ces blocs fera ainsi appel à une **indirection**. Si le fichier dépasse 138 blocs, la douzième entrée de l'«i-node» considéré permettra d'adresser 16 K-blocs supplémentaires, au moyen de deux indirections donc (128 blocs intermédiaires de 128 entrées chacun). Si ceci n'est pas suffisant, la treizième entrée permettra d'adresser 2 M-blocs de plus, au moyen de 3 indirections (128 fois 16 K-blocs, voir figure 5g). La taille du bloc était initialement de 512 octets, elle a tendance à croître avec l'évolution technologique (la capacité des disques): 1024 octets sur Unix System V et Berkeley 4.1 BSD, 4096 sur Berkeley 4.2 BSD;

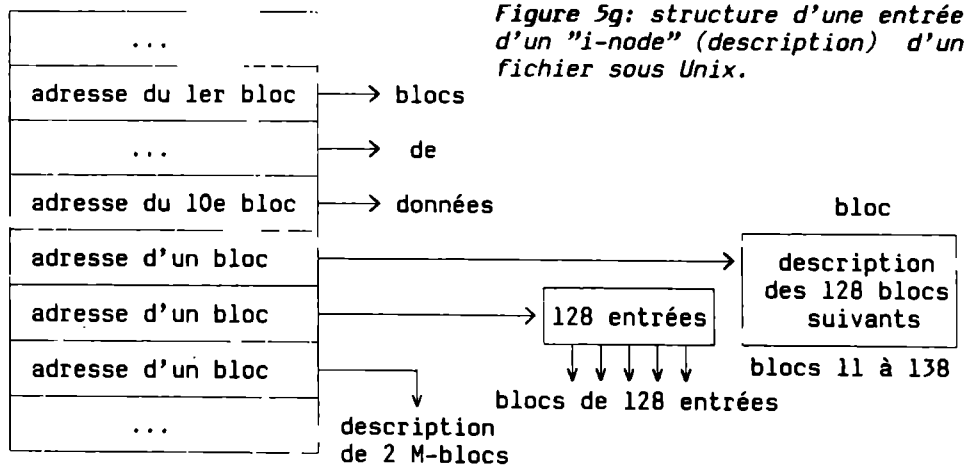


Figure 5g: structure d'une entrée d'un "i-node" (description) d'un fichier sous Unix.

- de façon totalement décentralisée, sans table, les différentes por-

BIBLIOTHÈQUE
Université du Québec à Rimouski

tions de disque allouées étant chaînées (linked) entre elles. C'est le cas de FLEX9, les deux premiers octets de chaque secteur contiennent respectivement les numéros de piste et de secteur du secteur suivant. Dans Tripos cette décentralisation va jusqu'à remplacer le répertoire (la liste des fichiers) par une table de "hash-code" (cf. chapitre 7). Dans MS-DOS le chaînage des "clusters" d'un même fichier est réalisé dans la FAT, d'où 4 Kclusters par fichier au maximum puisqu'une entrée de la FAT comprend 12 bits.

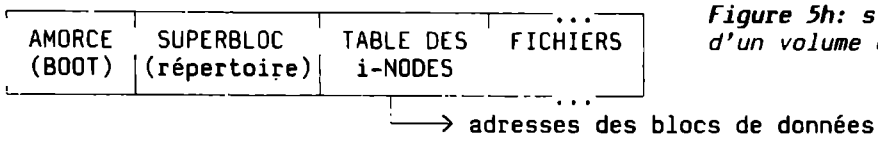


Figure 5h: structure d'un volume Unix.

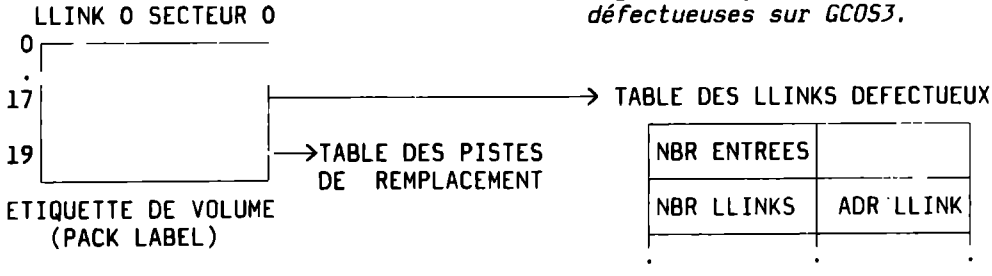
5.2.5 – Gestion de l'espace défectueux.

Un disque n'est pas toujours sans défaut, certains secteurs, certaines pistes, peuvent être défectueux et inaptes au stockage des données, soit dès la sortie de l'usine, soit en cours d'utilisation. Ces portions défectueuses ne doivent pas être attribuées.

Dans les méthodes centralisées il suffit de marquer occupées les entrées des tables associées aux zones défectueuses, s'il y a des tables d'allocation (entrée de la FAT égale à FF7 dans MS-DOS). Certains systèmes gèrent une table spéciale pour contenir la liste des portions défectueuses: cas de Cortex de CSEE, de GCOS3; elle doit être suffisamment grande pour pouvoir stocker toutes les zones défectueuses.

Dans les méthodes décentralisées, il suffit de ne pas inscrire ces portions défectueuses dans la liste des zones disponibles pour qu'elles ne soient jamais attribuées.

Figure 5i: gestion des pistes défectueuses sur GCOS3.



5.2.6 – Choix du disque.

Le disque (ou les disques dans le cas de fichiers multivolumes) sur lequel sera stocké le fichier peut être choisi par:

- l'utilisateur: il indiquera par exemple l'adresse physique de l'unité de disque. C'est le rôle du paramètre UNIT=371 ou VOL=SER=371 dans la commande // DD des OS d'IBM ([SAS86] parle de

"demand request"); la référence à l'unité peut être symbolisée:

VOL=REF=AUTRE.FICHIER

demande le volume contenant le fichier AUTRE.FICHIER,

VOL=REF=*.CMD

correspond au volume défini dans la commande (carte de contrôle dans le jargon IBM) //CMD. Dans la même commande, le paramètre AFF permet à l'utilisateur de forcer le choix en demandant l'unité utilisée pour tel autre fichier. De même, le paramètre SEP pour les séparer;

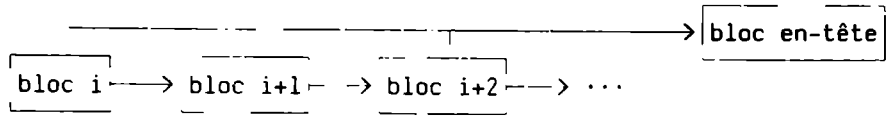
- le **système d'exploitation**: il choisira l'unité de disque selon divers algorithmes. On trouvera ainsi la méthode classique du **"tourniquet"**: la première requête est dirigée sur le premier disque, la seconde vers le second disque, etc., si la n-ième requête est dirigée sur le dernier disque, la (n+1)-ième requête le sera vers le premier disque, etc.; les unités de disque sont ainsi rangées dans une liste circulaire et l'allocation tourne à l'intérieur de ce cercle avec comme objectif de solliciter également chacun des disques. D'autres algorithmes sont basés sur une appréciation plus dynamique de la charge réelle: le disque le moins rempli, le disque le moins chargé (le moins accédé), etc. Pour les fichiers temporaires, le choix relève souvent du SE. Dans le cas des demandes "génériques" (selon [SAS86]: par modèle de disque, 3380 par exemple) la macro SCHEDULR de MVS permet de classer les modèles dans une liste parcourue lors de chaque allocation, créant ainsi une priorité d'allocation entre les modèles, le classement par défaut étant déduit des caractéristiques des disques;
- l'**utilisateur** et le SE conjointement, chacun contribuant au choix de l'unité de disque:
  * dans le cas des fichiers "catalogués", le choix est dirigé par le répertoire, cas fréquent sur les mini-ordinateurs avec répertoires arborescents (cf. chapitre 7) inspirés de ceux de Multics et GCOS3;
  * l'utilisateur donne un nom de type de disque, le SE choisira un disque dans l'ensemble des disques de ce type. C'est le rôle du paramètre UNIT dans les commandes // DD: UNIT=3330-1 conduira SRM à choisir un disque de modèle 3330-1 dans le cas de MVS, modèles définis par la macro IODEVICE lors de la génération du système;
  * l'utilisateur indique le nom d'un groupe d'unités de disques, nom défini lors de la génération ou du démarrage du SE. UNIT=SYSDA en est un exemple classique dans les commandes // DD, SYSDA est défini par la macro UNITNAME lors de la génération.

5.2.7 – Sécurité, confidentialité.

Afin d'assurer une meilleure disponibilité (cf. tome I, chapitre 19), le SE met en oeuvre différentes techniques. Certaines relèvent de la **redondance**:

- dans le cas des tables centralisées, on doublera la table: cas des structures de contrôle sur les disques de GCOS3, cas de la FAT de MS-DOS qui sont dupliquées;
- dans le cas des listes chaînées de zones disponibles, la gestion de l'espace est très sensible aux pertes de pointeurs (adresse de la

zone suivante) et on utilisera un double chaînage, avant et arrière;
- ajout de l'identification du fichier dans chacune des unités allouées. Cette identification pourra être matérialisée par un pointeur vers un bloc d'en-tête du fichier (cas de GCOS3, de Tripos père d'Amigados):



Au niveau des données, une sécurité supplémentaire conduit à écrire des informations redondantes de contrôle, en plus des codes détecteurs/correcteurs d'erreur (cf. tome IV) mis en place par le matériel du sous-système périphérique. C'est ainsi que GCOS3 ajoute des "checksum" dans les répertoires et certains fichiers comme les modules SSA, que MS-DOS en met un dans le secteur d'amorçage (boot).

Un autre aspect de la sécurité concerne la confidentialité: lorsqu'un utilisateur se voit attribuer de la place sur un disque, il est susceptible d'y retrouver l'information laissée par le fichier auquel la place avait été attribuée précédemment. Ainsi sous GCOS3 la commande $ FILE IN,X3P effacera le contenu du fichier IN, grâce au P de X3P, lorsqu'il sera rendu au SE en fin d'étape. La commande SET FILE/ERASE de VMS permet de même d'effacer les zones allouées en y écrivant des "profils d'effacement" (erasure pattern). Un autre dispositif de VAX/VMS permet d'interdire à l'utilisateur de lire au-delà du dernier bloc qu'il a écrit, dans la zone qui lui a été allouée: rôle du "qualifieur" /NOHIGHWATER. Sur une bande magnétique il est généralement interdit de lire après une écriture.

DOS/VSE supporte des fichiers "sécurisés" pour lesquels un message est envoyé au pupitreur lorsqu'on les ouvre (paramètre DSF de la commande // DLBL). Ce dispositif est généralisé dans les alarmes de VMS, associées aux listes d'accès (ACL) des fichiers (cf. chapitre 8).

5.3 – Allocation de l'accès aux disques.

Nous venons de voir comment est attribué l'espace sur les disques. Il convient aussi d'évoquer comment les processus obtiennent l'accès aux disques au cours de leur exécution, comment est alloué le "temps" disque. En dehors du cas des disques privés, disques réservés à un seul travail, à une seule étape à la fois donc, les disques sont partagés entre plusieurs étapes qui y accèdent concurremment. Quand un processus émet une requête vers un disque, le disque n'est pas forcément disponible, mais peut être en train de satisfaire la requête d'un autre processus. La nouvelle requête sera alors mise en attente (mise en "file d'attente", queued) jusqu'à ce que le disque concerné redevienne disponible (figure 5j).

Il y a donc allocation dynamique du temps disque. Cette allocation est réalisée par une partie du noyau, le gestionnaire des entrées/sorties physiques (GESP dans ce tome, IOS pour Input/Output Supervisor dans de nombreux SE).

Figure 5j: mise en attente des requêtes disque.

5.4 – Allocation de la mémoire centrale.

Comme la gestion de la mémoire centrale fait l'objet d'un chapitre particulier, nous n'en donnerons ici que les grandes caractéristiques.

Historiquement l'allocation de la mémoire centrale a été **statique** et **contiguë**: pour charger une forme exécutable il fallait trouver une zone contiguë de mémoire. Dans certains SE comme les DOS et OS/MFT d'IBM, l'adresse de chargement était de plus imposée et fixe. On parlait alors de **partitions**.

Certains SE ont permis des **extensions dynamiques** de l'espace initialement alloué qui restait cependant contigu (fonction GETMAIN des OS d'IBM, primitive MME GEMORE sur GCOS3). Le parallèle avec l'allocation de l'espace sur disque s'impose. L'inverse, c'est-à-dire le retour au SE d'un morceau de mémoire centrale, est aussi possible (fonction FREEMAIN des mêmes OS d'IBM). MS-DOS a introduit dans la version 2.0 les fonctions 48 et 49 de l'interruption 21 qui correspondent respectivement aux demandes d'allocation et de libération d'une zone mémoire.

Le **va-et-vient** (swapping) a ensuite permis de vider temporairement des étapes sur disque afin de faire de la place pour des étapes plus prioritaires ou afin de permettre à d'autres de s'étendre.

Aujourd'hui de nombreux SE allouent la mémoire centrale de façon **dynamique**, sous forme de quantum appelé **cadre**, sans nécessité de contiguïté.

La taille disponible en mémoire centrale est gérée en utilisant des tables d'allocation, des listes de zones disponibles, tout comme pour les disques.

Figure 5k: types de gestion de MC.

La confidentialité peut être améliorée en effaçant le contenu de la zone mémoire occupée par une étape qui se termine. GCOS3 est dans ce cas; dans NCP, logiciel des frontaux 37X5 d'IBM, lorsqu'un tampon est rendu au pool centralisé, un paramètre permet de demander sa mise à zéro. Dans le cas de MVS un nouveau cadre (cf. chapitre 11) alloué à un espace d'adresses (AS) est remis à zéro. Multics aussi ne remet à zéro un cadre que lors de son allocation.

L'allocataire de mémoire centrale fait partie du noyau. Sous GCOS3 (CALC), il prend cependant la forme d'une étape, mais pour des raisons de place: le noyau ne pouvant s'étendre au-delà de 64 Kmots certaines portions sont implémentées sous forme d'étapes toujours présentes.

5.5 – Allocation du processeur.

Le processeur est, lui aussi, alloué par un élément du noyau, le **distributeur** (dispatcher, task scheduler, traffic controller, CPU scheduler, short-term scheduler). Certaines machines implémentent dans le matériel, au niveau de l'architecture externe (cf. tome I), tout ou partie de cette commutation de processus: l'iAPX286 (ou 80286) à 10 Mhz met moins de 17 microsecondes pour sauvegarder tous les registres, charger les registres du nouveau processus élu ainsi que ses quatre descripteurs de segment, et reprendre l'exécution.

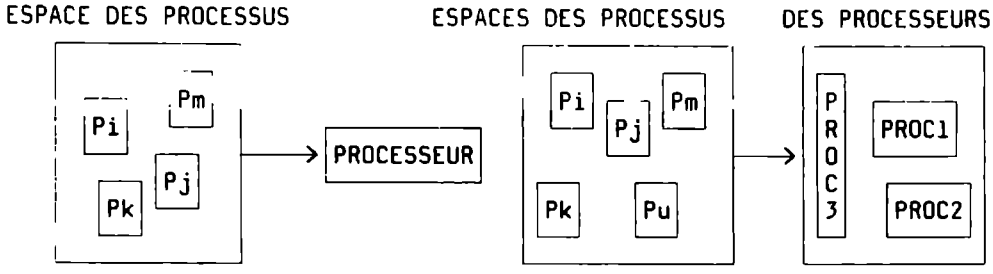


Figure 51: le distributeur dans sa fonction d'application entre l'espace des processus et l'espace des processeurs dans le cas d'un monoprocesseur et dans le cas d'un multiprocesseur.

5.5.1 – Priorités fixes.

Une des méthodes d'allocation les plus anciennes est basée sur des **priorités fixes**: chaque processus a une priorité fixe, il n'y a au plus qu'un processus par "niveau" de priorité. Les processus dans l'état prêt sont considérés comme formant une liste (ready queue) triée par ordre de priorités décroisssantes et le distributeur alloue le processeur au premier processus de cette liste. Quand le processus abandonne le contrôle du processeur, soit de son propre fait à la suite d'une demande d'entrée/sortie, d'une mise en sommeil temporaire par exemple, soit du fait d'un événement qui lui est externe comme l'arrivée d'une interruption, le distributeur donne la main au processus qui est à ce moment-là en tête de la liste des processus prêts.

Les processus de faible priorité ont peu de chance de s'exécuter et cette technique ne permet pas un taux de multiprogrammation très élevé: il n'est pas rentable de laisser en mémoire des processus qui ne sont jamais actifs. Il faut généralement donner les priorités élevées soit à des processus urgents, soit à des processus faisant beaucoup d'entrées/sorties: pendant que se déroulent les entrées/sorties, ils sont mis en attente (cas des entrées/sorties synchrones) et les processus de moindre priorité peuvent ainsi être actifs. Les processus les moins prioritaires seront généralement associés à des travaux peu urgents ou orientés calcul: ils sont toujours susceptibles d'utiliser le processeur.

Cette technique très **statique** d'allocation du processeur a long-

temps été la seule utilisée sur les systèmes DOS (Disk Operating System) d'IBM: elle fut aussi utilisée sous OS/MFT. Elle est implémentée dans iRMX86 (256 niveaux de priorité): lorsque deux processus y ont la même priorité, celui qui est prêt depuis le plus longtemps prend le contrôle du processeur, jusqu'à ce qu'une interruption et éventuellement un processus plus prioritaire reprenne le processeur.

Il faut bien distinguer cette **priorité d'exécution** (paramètre DPRTY des commandes // EXEC des OS d'IBM) prise en compte par le distributeur, des priorités de planification que nous avons vues plus haut (paramètre PRTY des commandes // JOB des OS d'IBM, paramètre "urgence", urgency, de la commande $ SNUMB de GCOS3).

5.5.2 – Tranches de temps.

L'équivalent de l'allocation **dynamique** par quantum de l'espace sur disque ou en mémoire centrale (pagination) est le **partage du temps** (time sharing) en quanta appelés "tranches de temps" (time slice). Les processus prêts sont considérés comme constituant une liste circulaire, et l'algorithme du tourniquet est appliqué: chaque processus reçoit une tranche de temps.

Cet algorithme est donc aux antipodes des priorités fixes puisque tous les processus sont considérés comme égaux en droits. Il a été introduit dans les premiers systèmes orientés vers le mode dialogué que l'on a d'ailleurs longtemps appelés environnements "temps partagé". Certains SE orientés vers le mode différé comme GCOS3 ont cependant une allocation du temps processeur basée sur les tranches de temps. Une des priorités de MFT et de MVT, systèmes à priorités fixes au départ, pouvait utiliser le mécanisme des tranches de temps (time slice option).

5.5.3 – Priorités fixes et tranches de temps.

Les deux techniques que nous venons de voir sont combinées dans certains SE: l'allocation se fait selon les priorités fixes, mais plusieurs processus peuvent avoir une même priorité. A l'intérieur d'un même niveau de priorité l'allocation se fait selon la méthode du partage de temps. Le distributeur donne d'abord le processeur aux processus de plus haute priorité et n'active les processus de priorité inférieure que lorsque les processus de priorité supérieure sont tous en attente.

On trouve cette méthode dans les SE qui veulent couvrir les domaines du temps réel et de l'interactif: **VMS** a 32 niveaux de priorité, 16 sont réservés aux processus "temps réel" (time critical, time dependent) et il n'y a qu'un processus par niveau de priorité; les 16 priorités inférieures (en fait les 15 car le niveau 16 est attribué au "swapper", le responsable du "swapping") sont attribuées aux processus interactifs, c'est-à-dire aux processus dialoguant avec les terminalistes (les utilisateurs connectés avec des terminaux interactifs) ainsi qu'aux processus des files "batch" simulées. Sur chacun de ces 15 niveaux, il y a attribution par tranches de temps. Les mesures de [CLA85] dans un environnement de développement avec une

trentaine de terminaux sur un VAX-11/780 donnent un changement de contexte toutes les 14 millisecondes environ, ce qui correspond à l'exécution de 5000 à 9000 instructions entre deux changements consécutifs.

RSX11M de DEC présente quelques analogies avec VMS, mais sans tranches de temps: il offre 250 niveaux de priorité, les 100 niveaux supérieurs sont réservés aux processus "temps réel"; sur les 150 niveaux inférieurs, les processus de même niveau sont gérés selon l'algorithme du tourniquet sans tranche de temps: le premier de la liste a d'abord la main, quand il se met en attente, le second prend la main, etc.

RTE-A sur HP1000, qui a des origines temps réel, offre 32767 niveaux de priorité: les priorités élevées (supérieures à la "fence") sont réservées aux processus temps réel; s'il y a plusieurs processus sur le même niveau de priorité, seul un d'entre eux s'exécute, les autres attendant sa terminaison ou sa suspension. Sur les basses priorités il peut y avoir plusieurs processus avec mécanisme de tranche de temps à l'intérieur d'un même niveau de priorité distribués comme dans VMS.

DMERT, version temps réel d'Unix ([H084]), répartit les processus en quatre types: le noyau, les processus noyau, qui représentent des applications temps réel, les processus superviseur et les processus Unix qui correspondent à des applications interactives. DMERT dispose de 16 niveaux d'exécution (16 priorités): les 13 niveaux supérieurs (3 à 15) sont réservés aux processus temps réel avec un système de priorités fixes, le niveau 2 est réservé au gestionnaire de la mémoire centrale et au distributeur des processus interactifs qui eux s'exécutent sur les deux plus bas niveaux de priorité, avec un mécanisme de priorités dynamiques, dites "logicielles", mécanisme lié aux caractéristiques E/S des processus ainsi qu'à la charge du système.

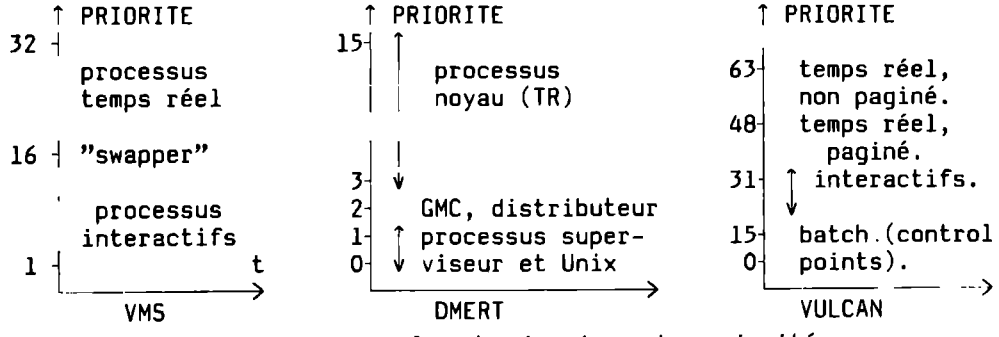


Figure 5m: exemples de structure des priorités.

HP-UX, extension d'Unix sur HP9000 ajoute au mécanisme de tranche pour les processus de faible priorité, une gestion de 255 priorités fixes. **Vulcan** de Harris dispose de 128 priorités logicielles (gérées par le distributeur) avec tranche de temps sur un même niveau. En interactif, les priorités sont associées aux terminaux.

Même les gros systèmes peuvent avoir des prétentions au temps réel: priorité A du distributeur de **GCOS3**. Ce SE permet aussi de définir des priorités intermédiaires en allouant plusieurs tranches de temps successives à un même processus.

5.5.4 – Priorités dynamiques.

Nous entrons dans un univers relativement complexe lorsqu'il s'agit de discuter des priorités dynamiques. Priorité dynamique signifie qu'un processus peut changer de priorité au cours de son exécution.

Prenons un algorithme de répartition du temps en tranches. Les processus faisant beaucoup d'entrées/sorties vont être défavorisés car ils n'épuisent pas toujours leurs tranches de temps: quand ils lancent une entrée/sortie ils sont mis en attente. Si l'on estime que les processus orientés entrées/sorties doivent être au moins aussi bien servis que les autres, parce qu'ils permettent d'atteindre une meilleure utilisation globale des ressources, il faut leur donner une priorité supérieure.

C'est ainsi que dans **GCOS3**, si l'on active l'option "I/O priority" du distributeur, les processus orientés E/S verront leur priorité augmenter en cours d'exécution. A intervalles réguliers, le distributeur de GCOS3 calcule pour chacun des processus prêts le rapport (temps E/S)/(temps processeur) et augmente la priorité de ceux pour lesquels ce rapport est élevé. Plus ils seront servis, plus ce rapport baissera. GCOS3 a en fait quatre niveaux de priorité associés:

- à la classe A pour les travaux de type temps réel,
- aux "courtesy call", mécanisme propre à GCOS3,
- à la classe B, généralement utilisée pour les sous-systèmes interactifs comme TSS, TDS, TPS. La priorité B se définit par le nombre de tranches qui peuvent être données aux autres travaux entre deux tranches consécutives données à la classe B,
- aux utilisateurs "normaux" avec 64 autres niveaux de priorité internes,

plus deux options (priorité E/S et priorité aux travaux "urgents") pouvant être combinées.

Dans les SE orientés vers le temps partagé on trouvera une préoccupation analogue concernant l'interactivité: le SE cherchera à favoriser les comportements **interactifs**, c'est-à-dire les utilisateurs se trouvant réellement sur des terminaux (par opposition aux travaux par lot simulé), et, parmi ceux-ci, à ceux qui dialoguent beaucoup avec le SE, donc lancent des commandes peu consommatrices de processeur. Le distributeur d'**Unix** calcule ainsi toutes les secondes les rapports (temps processeur)/(temps écoulé) de chaque processus et attribue le processeur à ceux dont le rapport est le plus faible. **GCOS64-E** dispose de 10 priorités susceptibles de varier en fonction du même rapport (temps processeur)/(temps écoulé).

VAX/**VMS** utilise aussi une technique de priorités dynamiques sur les niveaux associés aux processus interactifs, afin d'avantager les processus dont l'exécution est perturbée par l'environnement: chaque processus a une priorité de base en dessous de laquelle il ne descend

jamais. Si, lorsqu'il est actif, il ne peut aller au bout de sa tranche de temps (définie par le paramètre QUANTUM de VMS), sa priorité va augmenter d'une quantité liée à l'événement perturbateur. Cette priorité ne peut cependant dépasser la priorité maximum des processus interactifs. Chaque fois que le processus utilise pleinement sa tranche de temps, sa priorité est diminuée, sans bien sûr descendre en dessous de sa priorité de base. Le distributeur de CP (VM d'IBM) agit de même vis-à-vis des machines virtuelles qui bénéficient de l'option "exécution favorisée" (favored execution). DOS/VSE dispose maintenant d'une option "d'équilibrage dynamique des partitions" qui réattribue régulièrement les priorités de groupes de processus (partitions) afin que tous aient globalement la même utilisation du processeur.

Un des premiers SE interactifs, CTSS, disposait d'un algorithme dit "exponentiel" dans lequel la priorité d'une requête avait une valeur élevée au départ, puis décroissait rapidement si elle était longue à exécuter.

Multics, Primos, TOPS10 ont une technique basée sur l'existence de plusieurs listes de processus prêts, chaque liste correspondant à un niveau de priorité différent. Quand un utilisateur frappe une commande, son processus se trouve dans la liste de priorité élevée et il reçoit un quota de temps associé à cette liste. Le but est de faire en sorte que les commandes légères en consommation processeur (liste d'un répertoire par exemple) soient effectuées rapidement afin d'avoir un bon temps de réponse, objectif réaliste avec des commandes légères. Quand le processus a épuisé, en une ou plusieurs distributions successives, le quota initialement attribué, il passe dans la liste de priorité inférieure. Il en va de même dans VM d'IBM pour l'allocation du processeur réel aux machines virtuelles ([KOE85], [TET84]) qui passent successivement par les files exécutables (run lists) Q1, Q2, puis Q3.

MVS dispose de 256 priorités et donne le contrôle du processeur dans l'ordre suivant:

1. aux sorties spéciales (special exit) en premier,
2. aux tâches de type SRB qui ont une priorité "globale" (supérieure à celle de tout AS),
3. aux AS prêts, dans l'ordre des priorités contenues dans les ASCB qui forment une liste des AS prêts, chaque AS correspondant à une étape pour les travaux soumis par lots:
   3a. aux tâches de type SRB avec priorité locale pour cet AS d'abord,
   3b. aux tâches de type TCB enfin.

S'il n'y a aucune tâche prête, le mot d'état d'attente (wait PSW) est chargé (il existe en fait un ASCB de "WAIT"). C'est SRM qui peut changer les priorités ("chapping") des TCB pour adapter la charge aux objectifs de l'installation considérée et à l'état des ressources. Certains niveaux de priorité peuvent être définis comme APG (Automatic Priority Group) qui permet de définir trois groupes de priorité:

- un groupe MTTW (Mean Time To Wait) dans lequel la priorité est basée sur le temps moyen d'exécution avant de passer dans l'état en attente. Les AS rendant vite la main ont ainsi une priorité élevée

dans ce groupe. L'objectif est d'accroître le débit en favorisant le parallélisme entre l'activité du processeur et celle des E/S, but analogue à celui de la priorité E/S de GCOS3;
- un groupe à priorité rotative (rotate priority) où aucun AS ne peut dominer, la priorité étant périodiquement redéfinie dans ce but. Ces priorités sont donc gérées de façon analogue aux tranches de temps et sont supérieures à celles du groupe MTTW;
- les priorités fixes définies pour différentes périodes de la vie du travail ou des transactions. Elles sont inférieures aux priorités MTTW.

Notons que le pupitreur peut modifier la priorité d'un processus, par une commande (OCL): PRTY sous DOS/VSE.

Les priorités gérées par le distributeur sont parfois traitées de "logicielles" par opposition à celles qui sont associées aux interruptions et appelées "matérielles". C'est ainsi que tel processeur aura n "niveaux" d'interruption (cf. tome III). Les niveaux des interruptions ne sont pas nécessairement pris en compte au niveau du distributeur au même titre que les priorités des processus: dans certains cas les interruptions sont systématiquement plus prioritaires et sont prises en compte dès qu'elles arrivent, quelque soit la priorité du processus en cours.

5.5.5 – Cas des multiprocesseurs.

Nous distinguerons deux grands cas déjà présentés dans le chapitre 17 du tome I:
- dans certains multiprocesseurs, il y a un processeur **maître**, un ou plusieurs processeurs **esclave** et seul le processeur maître peut dérouler le code du distributeur. Il répartit le travail entre les processeurs. C'est le cas des biprocesseurs AP d'IBM, du VAX-11/782;
- dans d'autres multiprocesseurs, le distributeur peut s'exécuter dans tous les processeurs, chacun des processeurs allant se servir dans la liste des processus prêts. GCOS3, OS1100 fonctionnent de cette façon.

Il y a dans ce second cas un problème d'accès à la liste des processus prêts: un seul processeur peut à un moment donné en extraire un processus, les autres doivent attendre leur tour. La liste des processus prêts est un cas de **ressource critique** selon [CRO75], ressource à un seul point d'accès. Il faut donc des mécanismes capables de **sérialiser** les accès à ce type de ressource que nous présentons dans le chapitre 14.

5.6 – Allocation des canaux.

Le processeur central et la mémoire centrale (cf. tome I) sont reliés aux périphériques (disques, bandes, imprimantes, etc.) par des chemins et des unités de commande comme les canaux, les processeurs d'entrée/sortie, les "contrôleurs" de périphériques (cf. tome III). Ces ressources intermédiaires entre PC/MC et les périphériques sont allouées **dynamiquement** par le gestionnaire des entrées/sorties physiques (GESP) que nous avons introduit plus haut. Pour favoriser des

processus prioritaires (ceux des applications interactives par exemple) il peut se révéler intéressant de leur réserver des canaux afin qu'ils ne soient pas perturbés par les autres travaux moins prioritaires. [SCH80] recommande ainsi de séparer les canaux en en réservant certains pour des applications prioritaires comme le transactionnel.

Le gestionnaire des ressources SRM de MVS choisit le canal logique le moins utilisé, puis dans le cas des disques, celui qui a le plus petit nombre de fichiers alloués.

Certains SE permettent à l'utilisateur de demander que les E/S liées aux fichiers designés passent par des chemins différents (paramètre SEP des commandes // DD des OS d'IBM).

5.7 – Le noyau vu de l'extérieur.

Fonctionnellement le noyau peut être considéré comme un ensemble de fonctions appelées par des **primitives** codées dans les programmes qui utilisent ses services. Il répond ainsi à des demandes formulées par les utilisateurs, les primitives prenant la forme d'appel de sous-programmes (CALL) ou d'appel au superviseur (SVC sur 370, INT avec MS-DOS). On trouve dans [WIT84] l'expression "collection de fonctions disponibles que les utilisateurs peuvent inclure de façon sélective". On y trouve aussi la représentation du noyau comme la racine du recouvrement (overlay) global constitué par l'ensemble des séquences de code exécutées.

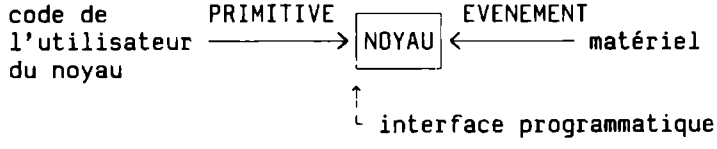


Figure 5n: noyau vu de l'extérieur.

Le noyau doit encore prendre en compte les événements (les interruptions) créés par les différents composants matériels de la machine.

Vu du noyau il y a ainsi deux types de séquences d'instructions qui sont déroulées:

- celles appartenant à des processus et activées par le distributeur selon les règles de la commutation de processus,
- celles déclenchées par les interruptions qui peuvent (cas des DPS7, iAPX286/386) ou non avoir une apparence de processus.

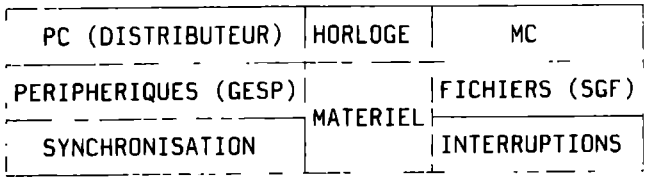


Figure 5o: structure générale du noyau.

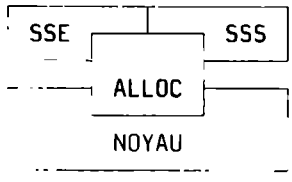
Structurellement le noyau peut être vu comme une entité particu-

lière (cas de celui de GCOS3) ou être lui aussi composé de processus soumis aux mêmes règles que les processus utilisateurs (cas de GCOS7), avec des privilèges supérieurs si nécessaire.

5.8 – Résumé, structure des SE.

Nous en sommes arrivés à un point où le système d'exploitation apparaît comme composé:

- d'un sous-système de gestion des travaux,
- d'un superviseur ou noyau chargé entre autres de la gestion des processus (task management). Plus généralement, le noyau gère:
  * le processeur: rôle du distributeur,
  * la mémoire centrale,
  * les interruptions: rôle du gestionnaire des interruptions GINT (interrupt handler IH),
  * les entrées/sorties physiques: rôle du GESP,
  * le temps: les processus peuvent demander à être réveillés à telle heure, dans n minutes, etc.



SSE : sous-système d'entrée
SSS : sous-système de sortie
ALLOC : allocataires de ressource
SSE+SSS: gestion des travaux

Figure 5p: structure globale du SE.

chapitre 6

gestion des entrées/sorties

6.1 – Les différents types d'entrées/sorties.

Le noyau du système d'exploitation est responsable de la gestion des entrées/sorties physiques, c'est-à-dire du transfert des **blocs** (block) entre mémoire périphérique MP et mémoire centrale MC. Dans les ordinateurs ayant au moins deux modes de fonctionnement (modes maître et esclave présentés dans le tome I) seul le noyau peut exécuter les instructions d'entrée/sortie.

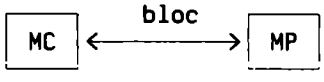


Figure 6a: rôle des entrées-sorties physiques.

MC: mémoire centrale; MP: mémoire périphérique

Les entrées/sorties physiques ne sont pas les seules opérations d'entrée/sortie dont on parle dans le logiciel: lorsque les applications (les programmes des utilisateurs) font des E/S, en utilisant les instructions classiques (READ, WRITE, GET, PUT, PRINT, INPUT, etc.) des langages de programmation de haut niveau comme Fortran, Cobol, Basic, PL/I, ils font ce que l'on appelle des E/S **logiques**, c'est-à-dire des E/S qui se déroulent totalement en mémoire centrale (figure 6b). Ces E/S ne concernent plus des blocs, mais des **articles** (ou enregistrements, record) et sont réalisées par les **méthodes d'accès**.

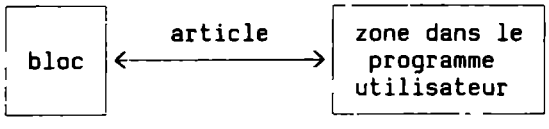


Figure 6b: rôle des entrées-sorties logiques.

Les entrées/sorties **virtuelles** quant à elles sont associées aux mécanismes de gestion de certaines mémoires virtuelles. Elles sont aussi gérées par le noyau.

Les entrées/sorties **spoulées** (spooled), enfin, sont liées aux périphériques lents et gérées en dehors du noyau (figure 6c).

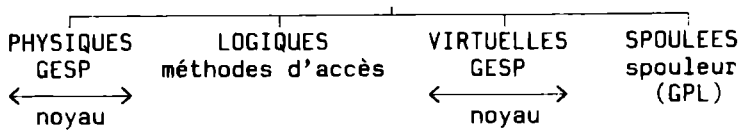


Figure 6c: les types d'E/S.

6.2 – Les entrées/sorties physiques.

L'entrée/sortie physique conduit à échanger un **bloc** entre le sous-système central et un sous-système périphérique, le bloc pouvant être réduit à un caractère. L'initialisation des E/S physiques se fait généralement au rythme du sous-système central (SSC) qui dispose des moyens de réguler le flot entrant par exemple: un périphérique ne peut pas de sa propre initiative envoyer des données que le SSC ne serait pas capable de prendre en compte correctement. Dans certains ordinateurs, les mini-ordinateurs en particulier, dans les environnements temps réel ou interactifs, certains périphériques peuvent, une fois activés, envoyer des données selon leur propre rythme.

La gestion des terminaux asynchrones (cf. chapitre 13) doit tenir compte de cette caractéristique. Une des solutions implémentées consiste à ranger les caractères reçus des terminaux dans une zone mémoire (un **tampon**, buffer, tumble table), rangement généralement réalisé automatiquement par le matériel (coupleurs, contrôleurs d'E/S asynchrones). Un processus du système vient régulièrement analyser les caractères accumulés et les distribuer aux différents processus destinataires. Comme la mémoire est limitée, ce tampon est utilisé de façon circulaire et il faut que le processus de distribution opère au moins aussi vite que les sources des données, sinon des caractères seront perdus. Situation qui peut se produire quand le système est surchargé et les mécanismes de priorité insuffisants. On peut disposer d'un tampon analogue utilisé dans l'autre sens, du SSC vers la périphérie, mais le contrôle de flux est plus facile à réaliser. D'autres solutions permettent de ne jamais perdre l'information en ralentissant la source, ce qui est concevable avec des terminaux interactifs, mais pas avec tous les appareils temps réel.

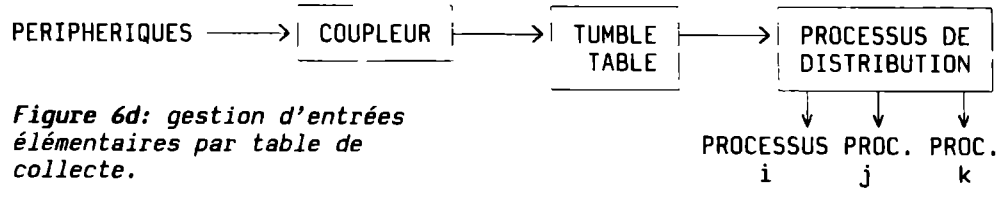


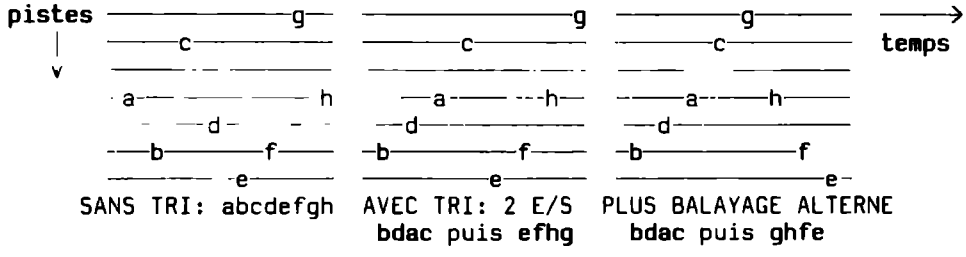
Figure 6d: gestion d'entrées élémentaires par table de collecte.

Sur les gros et moyens ordinateurs il n'y a généralement pas d'interaction au niveau du caractère entre un périphérique et un processus, les échanges se font par bloc et sur initiative du sous-système central.

Plus généralement le gestionnaire des E/S physiques GESP a comme fonctions:

- le **lancement** des entrées/sorties physiques, par une instruction

d'E/S par exemple (SIO sur 370, CIOC sur DPS8, etc.);
- le traitement des **interruptions** associées aux entrées/sorties, plus exactement la partie du traitement qui lui incombe et fréquemment appelée gestion de deuxième niveau des interruptions (cf. section sur la gestion des interruptions);
- l'analyse des conditions de fin d'E/S pour vérifier que tout s'est bien passé afin de lancer les actions prévues en cas d'incident: nouvelles tentatives, signalisation des erreurs insurmontables (unrecoverable errors), actions réalisées par les gérants d'appareil;
- ordonnancement des E/S en fonction des adresses: pour chaque disque certains GESP regroupent plusieurs demandes d'E/S physiques en une seule E/S (un seul programme canal au sens du tome III) afin d'optimiser le temps global pris par toutes ces entrées/sorties. Le gain s'opère surtout sur les mouvements de bras (seek) qui sont moins longs puisque le disque est parcouru en minimisant les retours en arrière à vide: ces retours ne se font qu'entre deux lancements, à moins que les requêtes ne soient alternativement regroupées en fonction des adresses croissantes puis décroissantes:



On constate le retard de l'E/S e dans le dernier cas à droite.

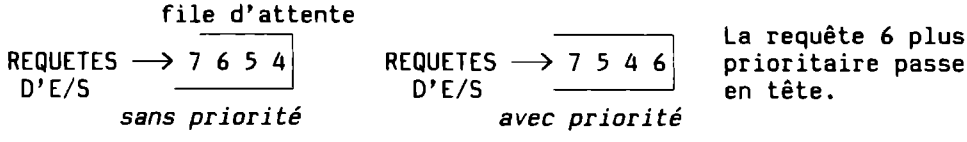
Le tri peut aussi se faire sur les adresses à l'intérieur d'une piste afin d'optimiser le délai rotationnel global. Il y a aussi un gain sur le processeur central, car lancer une E/S physique revient en effet à dérouler un grand nombre d'instructions, couramment plusieurs milliers sur les gros et moyens ordinateurs (tableau 6e). Pour optimiser au mieux, certains GESP ont tendance à retarder le lancement des demandes en attente, afin de lancer une E/S importante regroupant les futures demandes; cette technique a le défaut d'augmenter les temps de réponse. MVS a ainsi essayé plusieurs méthodes au long de ses différentes versions pour gérer les entrées-sorties concernant les pages [DUB86]: dans une première version les pages étaient écrites sur un disque tant que le **temps de service** (cf. figure 6g)) de l'E/S ne dépassait pas 50 ms, au-delà on changeait de disque; cet algorithme avait le défaut de charger les disques les plus rapides et des versions suivantes ont tenté **l'équilibrage** de charge en écrivant successivement sur les différents disques d'un même fichier de pagination (PDS); une troisième approche est basée sur la réduction de la **déperdition** (overhead), le

système n'a pas à construire des blocs de contrôle à chaque E/S, il maintient des E/S actives en permanence, ce qui conduit à dédier des canaux; le GESP doit alors utiliser les PCI (interruptions intermédiaires, voir tome III) pour réagir à l'interruption générée par le positionnement (seek) de la requête suivante: le transfert correspondant peut être exploité aussitôt, sans attendre la fin du programme canal (la fin de l'exécution de toutes les requêtes), mais les résultats ne sont toujours pas estimés satisfaisants;

MACRO SYSTEME	MVT	VS1	VS2.R3
LANCEMENT E/S	660	1250	5000
EXCP	630	1800	2110
WAIT	550	900	840
POST	520	700	700
GETMAIN	730	840	1250
FREEMAIN	660	800	790
TIME	720	730	730
STIMER	1150	660	1490
DEQ	790	950	1210
ENQ	1300	950	1160

Tableau 6e: nombre d'instructions déroulées par différentes macros sur différentes versions des OS d'IBM selon [LAM77].

- la mise en oeuvre de **priorités**: le fonctionnement habituel est de type FIFO (la première demande reçue est la première réalisée). Certaines demandes peuvent être prioritaires et mises systématiquement en-tête de la liste des demandes en attente: cas des E/S du SE sur GCOS3. Les demandes peuvent aussi être classées en fonction de la priorité du processus demandeur:



- le choix du **chemin** entre la mémoire centrale et le périphérique: nous verrons dans les tomes III et IV qu'il peut y avoir plusieurs chemins. Certains GESP utiliseront un algorithme de tourniquet sur les chemins gérés en liste circulaire. Dans d'autres le choix se fera toujours en commençant par le premier chemin (cas de GCOS3);

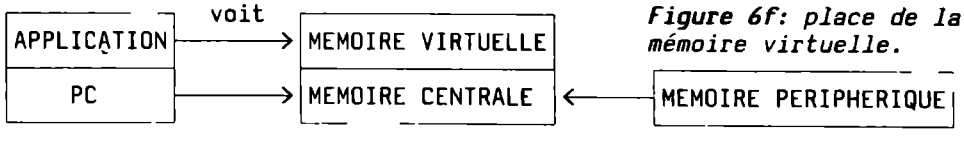


Figure 6f: place de la mémoire virtuelle.

- la **traduction** des adresses des tampons dans les programmes canal lorsque le SE utilise une mémoire virtuelle (absolutisation dans GCOS7 par l'instruction ACCE): les canaux, les processeurs

d'entrée/sortie travaillent en grande majorité avec des adresses réelles: certains compatibles 370, des DPS7, ont des canaux travaillant avec des adresses virtuelles, d'où un gain de puissance processeur puisque la traduction des adresses n'est plus nécessaire;
- la fixation des tampons en mémoire centrale dans le cas des SE à mémoire virtuelle ou lorsque les programmes sont translatables (tome V) en mémoire centrale: il n'est pas question de déplacer une portion de programme alors qu'une E/S va y accéder en utilisant son adresse réelle (cf. traduction des adresses ci-dessus).

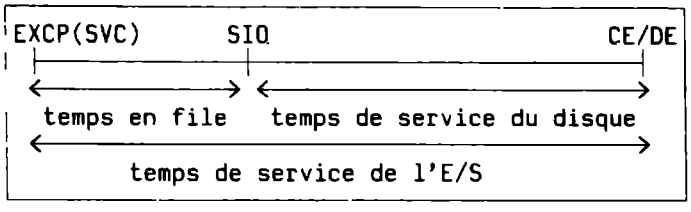


Figure 6g: modèle d'une E/S physique.

DE : Device End
CE : Channel End
SIO : Start I/O
EXCP: EXecute Channel Program

Les E/S physiques sont déclenchées à la suite des demandes des programmes d'application réalisées par une instruction de type SVC (SuperVisor Call, appel superviseur) dans les SE d'IBM de la série 360/370: elles provoquent une interruption qui active, dans le noyau, le GESP. Sous GCOS3 ce sera l'instruction MME (Master Mode Entry) qui jouera un rôle identique, ER (Executive request) dans l'OS1100, INT sous MS-DOS, etc. On trouvera aussi le niveau des macro-instructions comme EXCP (EXecute Channel Program chez IBM) qui est chargée de déclencher la SVC (figures 6g et 6h). Les termes de PIO, PIOCS, IOS, IOCS correspondent eux au gestionnaire des E/S physiques, son nom sur VMS est QIO (Queued InputOutput).

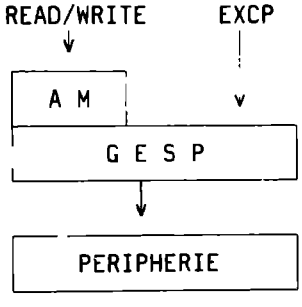


Figure 6h: E/S logiques et physiques accessibles aux applications dans un SE du type MVS.

AM: Access Method
GESP: gestionnaire des E/S physiques

6.3 – Entrées/sorties synchrones et asynchrones.

Pour les utilisateurs habitués aux langages de haut niveau, lorsqu'une instruction d'E/S est exécutée (une instruction du type READ, WRITE, etc.), l'instruction suivante n'est exécutée qu'après la fin du transfert, c'est-à-dire, dans le cas d'une lecture, lorsque l'information demandée est délivrée à l'utilisateur. Ainsi en Cobol la séquence:

```
READ FICHIER INTO ZONE.
DISPLAY ZONE.
```

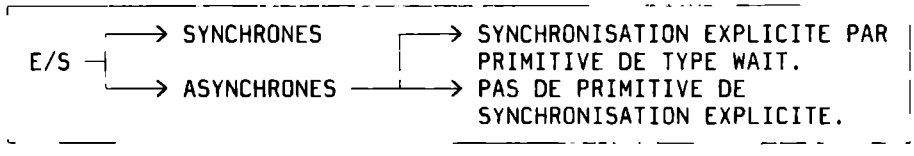
affichera le contenu de ce qui a été lu depuis le fichier dans ZONE. De même en Basic

```
10 LINE INPUT #1,A$
20 PRINT A$
```

affichera le contenu de A$ tel qu'il figure dans le fichier lu.

Ces E/S sont dites **synchrones** (ou bloquantes): l'opération d'entrée/sortie synchronise le programme utilisateur, PRINT Basic ne s'exécute qu'après la fin de l'E/S, idem DISPLAY en Cobol.

Lorsque l'E/S ne synchronise pas le programme utilisateur, l'instruction qui suit l'instruction de READ/WRITE/etc. est exécutée aussitôt, sans attendre la fin de l'E/S. Ce sera à l'utilisateur de tester si l'E/S est terminée (instructions du type WAIT ou CHECK) ou bien de demander à être averti de la fin de l'E/S par un mécanisme du type interruption: cas des AST (Asynchronous System Trap) de VMS.



L'instruction INKEY$ du Basic permet d'illustrer les E/S asynchrones:

```
10 A$=INKEY$
20 IF A$ <> "" THEN PRINT A$ ELSE PRINT "rien"
30 GOTO 10
```

affichera le caractère correspondant à la touche appuyée sur le clavier ou bien la chaîne "rien" si aucune touche n'est appuyée. En aucun cas le programme n'attendra un appui sur le clavier pour passer de la ligne 10 à la ligne 20. Si la ligne 10 contenait INPUT A$ il y aurait par contre attente jusqu'à appui sur la touche d'envoi (retour chariot). Les deux exemples ci-dessous illustrent les différences entre E/S asynchrones dans VMS:

```
TEMPS:.BLKQ     1
      etc.
      $SETIMR_S EFN=#33,-
                DAYTIM=TEMPS
      $WAITFR_S EFN=#33
      (...)
```

```
MIDI: .BLKQ     1
      .ENTRY    LIBRAIRIE,0
      $SETIMR_S DAYTIM=MIDI,-
                ASTADR=TEMPSAST
      BSBW      ERROR
      (...)
      .ENTRY    TEMPSAST,^M<>
      (...)
      RET
      .END LIBRAIRIE
```

L'exemple de gauche correspond à un fonctionnement asynchrone synchronisé: l'appel du service "horloge" de VMS par $SETIMR_S demande au SE de réveiller le processus au bout d'un temps défini par le contenu de la variable TEMPS. L'instruction $WAITFR_S permet de se synchroniser, d'attendre le réveil. Le nombre 33 qui figure dans le programme correspond à un numéro d'événement (event flag), mécanisme de synchronisation interprocessus utilisé par VMS.

L'exemple de droite illustre le fonctionnement asynchrone avec interruption (trap dans la terminologie VMS) sans primitive de synchronisation explicite. Dans cet exemple, le processus demande au SE de le réveiller à l'heure définie par la variable MIDI. Le réveil doit se matérialiser par l'exécution de la séquence TEMPSAST, jusqu'à l'instruction de retour RET. A MIDI, le processus utilisateur est interrompu, la séquence TEMPSAST est exécutée, puis le processus reprend son exécution à l'endroit où il a été interrompu. Un mécanisme analogue appelé "courtesy call" (appel de service) est implémenté dans GCOS3. Dans l'univers des PC un fonctionnement du même type est illustré par la réception d'un paquet dans le dialogue par flot (stream) sur le réseau local PCnet d'Orchid. Le Basic offre quelque chose d'approchant avec l'instruction ON (ON ERROR, ON KEY par exemple, pas ON I GOTO par contre) dans la mesure où une séquence est activée à la suite de l'arrivée d'un événement, une erreur dans le cas de ON ERROR. De même les ON CONDITION du PL/I.

Dans iRMX86 la couche de logiciel BIOS offre un service principalement asynchrone, la couche supérieure EIOS apporte un service synchrone.

6.4 - Entrées/sorties logiques.

Les E/S logiques se déroulent en mémoire centrale uniquement et concernent l'unité de données que manipule l'utilisateur: l'**article**. Le bloc concerné par les E/S physiques a pour but principal d'optimiser les mouvements d'information entre mémoire centrale et mémoires périphériques; il est aussi lié à un souci d'optimisation dans l'utilisation des surfaces de stockage (points développés dans les tomes III et IV, cf. figure 6i).

Les programmes des utilisateurs manipulent des articles qui se trouvent dans des fichiers; articles et fichiers sont des notions **logiques** alors que blocs, volumes sont des notions **physiques**. Quand un utilisateur manipule un article, cet article est généralement stocké dans une zone de travail (work area) gérée par l'utilisateur. Ce sera par exemple une structure Cobol:

```
01 MON-ARTICLE.
  02 NUMERO-PIECE PIC 999.
  02 PRIX-PIECE   PIC 999V99.
  02 STOCKAGE.
    03 NOMBRE     PIC 9(4).
```

Supposons que notre utilisateur soit en train de créer un fichier contenant l'état de toutes les pièces que fabrique la société où il travaille. Avec des instructions de traitement, des MOVE en Cobol, il

remplira chacun des champs de la structure MON-ARTICLE (MOVE "431" TO NUMERO-PIECE. par exemple) pour chacun des types de pièce fabriqués. Chaque fois qu'il aura ainsi rempli la structure pour une pièce, il écrira l'article dans le fichier, par une instruction d'E/S, l'instruction WRITE en Cobol.

article	⟶	traitement
bloc	⟶	mouvement MC ⟷ MP
secteur, zone	⟶	stockage
page	⟶	gestion MC

Figure 6i: relations entre unités de données et différentes notions.

Cette instruction WRITE ne déclenchera pas obligatoirement une E/S physique telle que nous l'avons décrite dans la section précédente. Elle déclenchera par contre à coup sûr une E/S logique qui consistera à copier le contenu de la zone de travail MON-ARTICLE dans le tampon associé au fichier concerné. Un tel tampon contiendra un bloc, c'est-à-dire plusieurs articles, et ce n'est que lorsque suffisamment d'écritures logiques auront été exécutées pour remplir le tampon que ce dernier sera mis sur mémoire périphérique par une E/S physique (figure 6e).

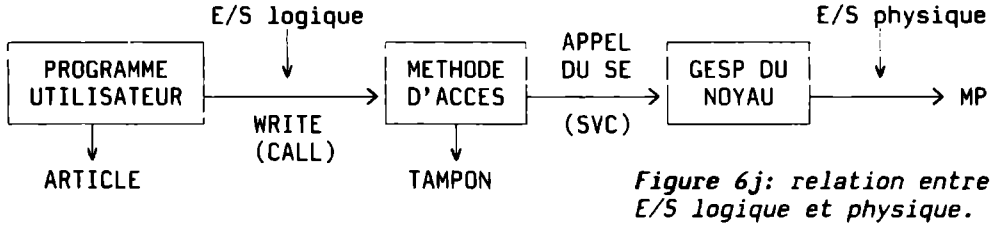


Figure 6j: relation entre E/S logique et physique.

Le logiciel responsable de cette copie entre zone de travail et tampon est généralement appelé **méthode d'accès** (AM, Access Method). On trouvera aussi le terme de LIOCS (Logical Input Output Control System) dans DOS/VSE. Les méthodes d'accès n'appartiennent pas au noyau et se présentent sous forme de sous-programmes appelés par le programme utilisateur:

- sous-programmes écrits par l'utilisateur qui veut disposer de sa propre méthode d'accès. Ils sont alors assemblés ou compilés avec le code de l'utilisateur;
- sous-programmes plus généralement produits par le fournisseur du SE et stockés dans des fichiers disque appelés **bibliothèques** ou librairies (library). C'est au moment de l'édition de liens qu'ils sont insérés dans le programme de l'utilisateur (cf. tome V);
- ces sous-programmes peuvent pour des raisons de performance (en temps et en espace) résider en mémoire et être dynamiquement liés aux programmes utilisateurs au cours de leur exécution. Ils doivent alors être ré-entrants (cf. tome V). C'est le cas de certaines méthodes d'accès de DOS/VSE qui se trouvent dans la zone mémoire appelée SVA (cf. chapitre 12).

Avec les méthodes d'accès on comprend mieux la difficulté à définir ce qui fait ou non partie du SE: n'importe qui peut ajouter une méthode d'accès. Nous considérerons ici que ces méthodes d'accès font partie du SE, comme le sous-système d'entrée par exemple, mais n'appartiennent pas au noyau. Il est d'ailleurs intéressant de noter qu'Unix est généralement réduit à un noyau (kernel) et à un interpréteur de commandes (le Shell); pour Unix un fichier est un ensemble de caractères, il n'y a aucune notion d'article, aucune méthode d'accès élaborée. Les fournisseurs de systèmes basés sur Unix ont dû ajouter les méthodes d'accès nécessaires pour offrir des systèmes complets.

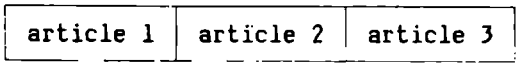


Figure 6k: un bloc de 3 articles.

Pour mieux comprendre la différence entre E/S logiques et E/S physiques, prenons un exemple numérique avec des tampons de trois articles (cf. figure 6f). On parlera de **facteur de blocage** égal à 3. Une première lecture du fichier (par l'instruction Cobol "READ fichier INTO MON-ARTICLE." par exemple) aménera l'article 1 dans la zone de travail MON-ARTICLE du programme de l'utilisateur. Comme nous supposons qu'au départ le tampon est vide, cette copie de l'article 1 entre le tampon et la zone de travail devra être précédée d'une "alimentation" du tampon, c'est-à-dire d'une entrée/sortie physique entre la mémoire périphérique (MP) et la mémoire centrale (MC). L'E/S logique déclenchée par l'instruction READ provoque ainsi le déroulement d'une E/S physique exécutée par le superviseur pour le compte de l'utilisateur.

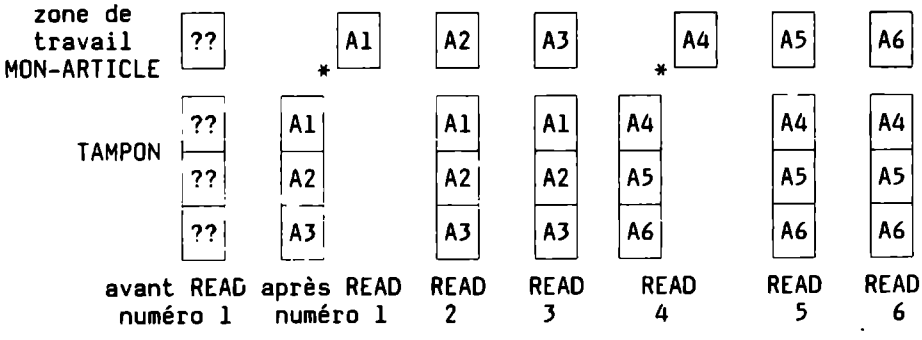


Figure 6l: évolution du tampon et de la zone de travail en fonction des lectures logiques.
(*: E/S physique, ?: non significatif)

La seconde lecture (READ) correspondra à une deuxième E/S logique qui amènera l'article 2 dans MON-ARTICLE. Il n'y aura aucune E/S physique associée. La troisième lecture amènera de même l'article 3 dans MON-ARTICLE, sans déclencher d'E/S physique. Par contre la quatrième

E/S logique (instruction READ du Cobol) provoquera, comme la première, une E/S physique qui alimentera le tampon avec les trois articles suivants, à savoir les articles numéros 4, 5 et 6 (figure 61).

On constate qu'il n'y a qu'une entrée/sortie physique pour trois entrées/sorties logiques. Or ce sont les E/S physiques qui sont très chères.

Dans certains cas (le lecteur de cartes sous GCOS6-MOD400 par exemple), l'ouverture de la ressource (verbe OPEN en Cobol) provoque une première lecture physique anticipée.

L'effet du facteur de blocage est illustré par le tableau 6m qui donne des chiffres obtenus sous MVS. Le passage d'une taille de bloc de 200 à 6000 octets fait diminuer le temps écoulé de plus de 90 %, la consommation de processeur baisse de plus de 87 %, le temps canal est réduit de 65 %.

TAILLE DU BLOC	200	1000	2000	4000	6000
NOMBRE D'EXCP	105000	21000	10500	5250	3500
NOMBRE DE SIO	51319	10468	5277	2728	1800
TEMPS ECOULE	600	150	103	71	56
TEMPS PROCESSEUR	74	22	14	11	9
TEMPS CANAL	95	47	38	37	33

Tableau 6m: résultats publiés par [SCH80] dans le cas de QSAM avec 5 tampons (BUFNO=5, temps en secondes).

6.4.1 – Modes "move" et "locate".

Nous avons vu dans le cas des entrées/sorties logiques qu'il y avait copie de chaque article entre le tampon associé au fichier et la zone de travail de l'utilisateur. C'est ce que l'on appelle des E/S logiques en mode "déplacement" (move mode). Cette copie correspond à une consommation de la ressource processeur qui est fonction de la taille de l'article. Il est possible d'éviter de consommer cette puissance processeur en ne recopiant pas l'article dans une zone de travail de l'utilisateur: l'E/S logique se limite alors à fournir à l'utilisateur un **pointeur** vers l'article dans le tampon, et l'on parlera alors de mode "localisation" (locate mode). Si l'on considère la déclaration suivante de fichier en Cobol:

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT MON-FICH ASSIGN TO JOUR.
DATA DIVISION.
FILE SECTION.
FD  MON-FICH
DATA RECORD IS MON-REC.
    01 MON-REC.
       02 NUM-PIECE     PIC 999.
       02 PRIX-PIECE    PIC 999V99.
       02 STOCKAGE.
          03 LIEU       PIC X(12).
          03 NOMBRE     PIC 9(4).
```

L'instruction "READ fichier." fera en sorte que la structure MON-REC soit utilisée comme une grille plaquée sur chaque article lu dans les systèmes offrant le mode "locate", cas des OS et DOS d'IBM. Ce mode "locate" présente une moindre sécurité dans la mesure où l'utilisateur

peut "déborder" de l'article courant et modifier directement d'autres parties du tampon, sinon d'autres tampons ou zones mémoire. VMS offre de même les deux modes "move" et "locate", ce dernier en lecture seulement à cause des problèmes de sécurité posés par l'écriture dans les tampons gérés par la méthode d'accès. RTE-A offre aussi un mode "move" qui permet de n'utiliser que le tampon de l'utilisateur, pour les fichiers de type 1 ou 0, ces derniers correspondant à des unités qui ne sont pas des disques.

L'exemple ci-dessous extrait de [DUB86] offre une illustration complémentaire des entrées/sorties: on y voit quatre "intervenants":
- le programme d'application ou programme utilisateur,
- la méthode d'accès à laquelle il fait appel,
- le gérant (driver),
- le superviseur du SE (GESP).

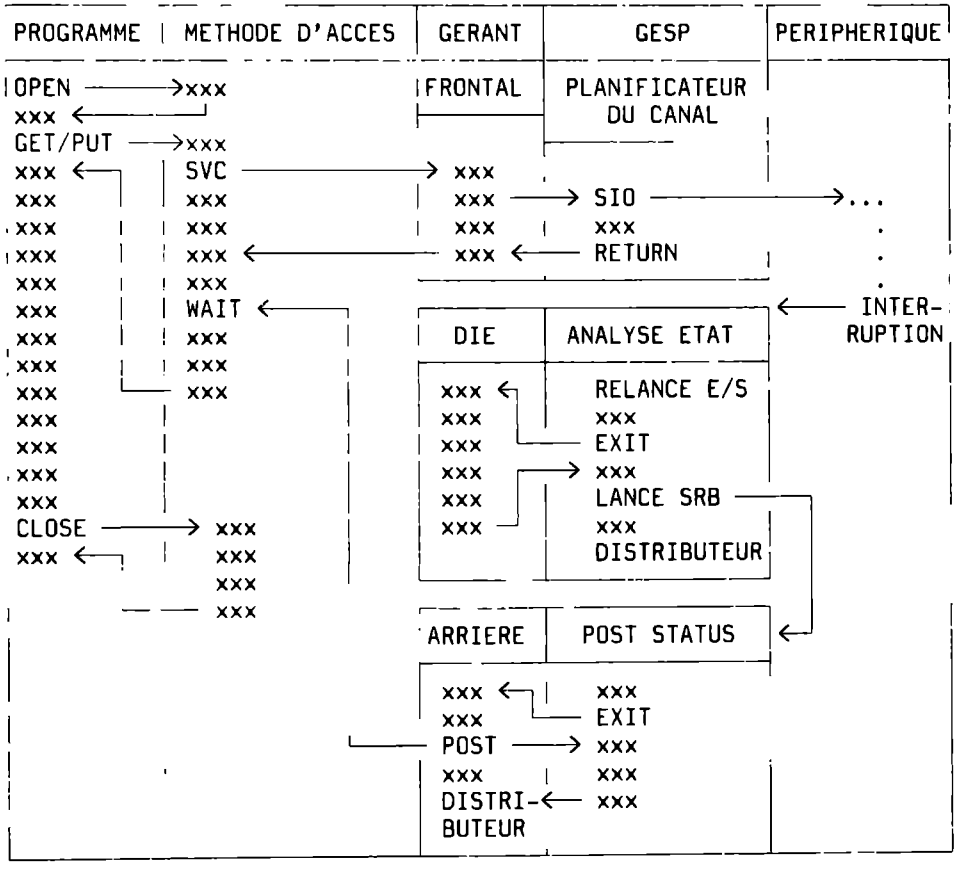


Figure 6n: E/S sous MVS selon [DUB86].

6.4.2 – Tampons multiples (multiple buffering).

Reprenons l'exemple de la figure 61 et supposons que:
- une E/S physique dure un temps **d** égal à 50 ms, vue du sous-système central;
- cette E/S physique requiert un temps processeur **p** égal à 3 ms pour la lancer et traiter l'interruption finale (cf. tome III);
- une E/S logique prenne un temps processeur **e** égal à 0,1 ms;
- le temps **t** de traitement d'un article par l'application, est égal à 2 ms.

En négligeant les temps d'ouverture et de fermeture de fichier, le traitement d'un fichier de **n** articles prendra un temps T égal à:

$$T = n*(e+t) + [(n+2)/3]*(p+d)$$

Pour n=3000 articles, nous aurons T=59,3 secondes, la consommation totale de temps processeur étant égale à n*(e+t) + [(n+2)/3]*p, soit 9,3 secondes. La figure 6o illustre les différentes phases du traitement au cours du temps. Nous constatons que le processeur est très peu utilisé (16 %), tout se passant de façon séquentielle, il n'y a pas de simultanéité entre l'activité du processeur et celle de la périphérie.

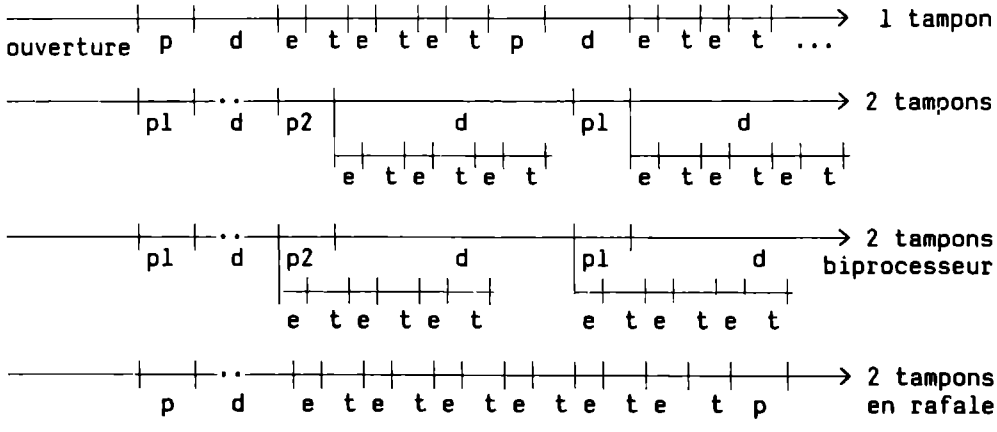


Figure 6o: modèle du traitement d'un fichier (pi=E/S physique, i=tampon, d=attente, e=E/S logique, t=traitement).

Une méthode qui permet de diminuer le temps de traitement du fichier en créant une simultanéité d'activité des ressources centrales et périphériques conduit à utiliser plusieurs tampons, un tampon étant rempli par une E/S physique pendant que l'autre satisfait les E/S logiques (cf. figure 6o). VMS parlera de "read-ahead" et de "write-behind" ([VAX84]).

Dans notre exemple le temps de traitement descend à [(n+2)/3]*(p+d) soit 53 secondes au lieu de 59,3. Le gain est peu important car nous avons affaire à une application limitée par les E/S (I/O bound) dans laquelle le temps d'E/S est plus de cinq fois supé-

rieur au temps processeur. Seule la multiprogrammation est susceptible ici de mieux utiliser le processeur. Un biprocesseur, l'un s'occupant des E/S physiques, l'autre de l'application, serait peu efficace et ne réduirait le temps de traitement du fichier que de 3 secondes (figure 6o). Certains systèmes utiliseront l'entrelacement des blocs ou secteurs sur disque (cf. tome IV), la distribution des blocs sur différents disques.

Le nombre de tampons associés à un fichier peut être défini à différents niveaux:

- au niveau du système d'exploitation: ce nombre joue le rôle de valeur par défaut,
- au niveau du langage de commande (paramètre BUFNO dans les OS d'IBM),
- au niveau du programme lui-même (clause "n AREAS" en Cobol). Le terme d'AREA utilisé dans RMS (méthodes d'accès de VMS) pour l'indexé séquentiel correspond à quelque chose de différent; l'«area» y est une sorte de sous-fichier avec ses propres caractéristiques, taille de "bucket", tailles de première allocation, d'extension, etc. Cette notion permet entre autres de répartir un même fichier sur plusieurs volumes différents afin d'améliorer les temps de traitement.

Dans le cas de MS-DOS, le paramètre BUFFERS=n du fichier CONFIG.SYS définit le nombre de tampons utilisés pour tous les fichiers ouverts. On parlera de "pool" de tampons. VMS ([VAX84B]) utilise le terme de "bloc" pour "secteur" (cf. tome IV), le terme de "multibloc" pour une E/S physique correspond à plusieurs blocs (de 1 à 127, nombre défini par la commande DCL SET RMS_DEFAULT/BLOCK_COUNT=n, 16 par défaut) d'un fichier organisé en séquentiel (cf. chapitre 7); le terme de "bucket" (1 à 63 blocs) y correspond à notre notion de bloc dans le cas des organisations relatives et indexées. L'expression de "tampons multiples" (multiple buffer) correspond dans VMS à une sorte d'antémémoire vis-à-vis du fichier, le nombre de tampons étant défini par la commande DCL SET RMS_DEFAULT/BUFFER_COUNT=n. Dans les méthodes d'accès comme UFAS sous GCOS, VSAM chez IBM, le bloc, le "bucket", sont appelés CI (Control Interval):

TERMINOLOGIE UTILISEE DANS CE TOME	VAX/VMS	COBOL	VSAM	UFAS
secteur	bloc			
bloc	multibloc ou bucket	area	CI	CI
sous-fichier	area			

6.4.3 – Tampons multiples en rafale.

Des méthodes d'accès comme UFAS (Unified File Access) de GCOS3 utilisent les tampons multiples d'une façon particulière, les k tampons étant transférés par une seule E/S physique. Ceci revient à traiter des super-tampons de 3 k-articles dans notre exemple, en ne transférant qu'un super-tampon à la fois. Dans ce contexte on cherche à réduire le coût processeur et le temps total d'entrée/sortie au détri-

ment du temps de traitement de fichier. La simultanéité entre le traitement et les entrées/sorties est confié à la multiprogrammation. Le temps de traitement de notre fichier passera avec 2 tampons à 33,8 secondes (contre 53) en considérant que le temps d'une E/S physique passe de 50 à 52 ms, le temps processeur passant de 9,3 à 7,8 secondes; comme le programme est orienté E/S, cette méthode améliore son temps de traitement.

SAM-E de MVS utilise aussi cette technique avec 5 tampons par défaut à chaque E/S physique, rendant obligatoire le "chained scheduling" que l'on peut préciser lors de l'ouverture du fichier avec QSAM. QSAM essaie (cf. [SCH80]) aussi de regrouper le maximum de tampons par E/S physique, mais écrit les tampons prêts dès que le périphérique est disponible. Ainsi dès que le premier tampon est plein, QSAM va l'écrire pendant que l'utilisateur remplit le second tampon. Si le disque est toujours occupé quand ce second tampon est plein, ce tampon ne sera écrit sur disque que plus tard. Supposons que le disque devienne libre quand le troisième tampon est plein, ils seront chaînés par QSAM ("chained scheduling"). En moyenne deux blocs seulement seront écrits lors de chaque E/S physique d'après les mesures réalisées.

RTE-A utilise un tampon minimum de 256 octets (la taille du secteur de base sur disque) que l'utilisateur peut augmenter jusqu'à 127 fois.

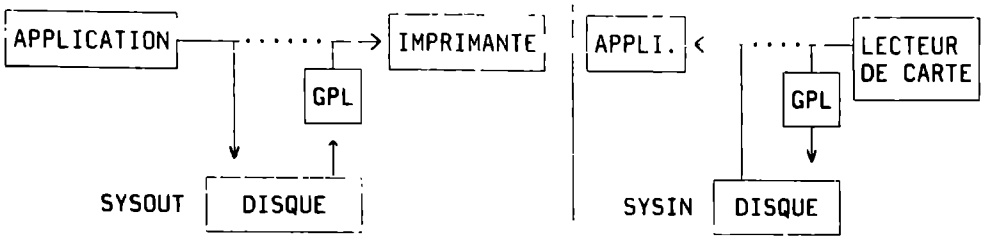
6.5 – Gestion des entrées/sorties lentes.

6.5.1 – SYSIN, SYSOUT, spoule.

Un processus qui traite des fichiers peut être ralenti par la vitesse des périphériques supportant ces fichiers. Nous l'avons illustré dans notre exemple ci-dessus lorsque nous avons introduit deux tampons. Ce ralentissement ne peut qu'être constaté et ses effets limités par la multiprogrammation dans le cas des périphériques rapides comme les disques ou les bandes. Il devient très gênant lorsque l'on a affaire à des périphériques intrinséquement lents comme les imprimantes, les lecteurs de cartes car alors les processus "traînent" longtemps en mémoire centrale, la ressource la plus coûteuse après le processeur.

Dans le cas de périphériques lents comme les écrans interactifs on peut vider le processus (swapping) sur mémoire secondaire (sur disque) pendant que l'on attend un message du terminal.

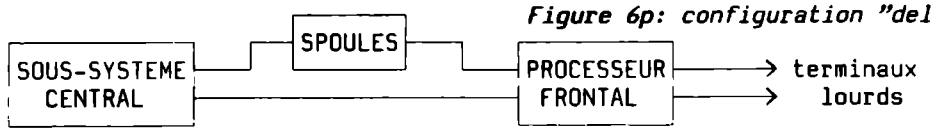
Dans le cas des périphériques unidirectionnels comme les imprimantes ou les lecteurs de cartes on simulera ces périphériques sur un appareil plus rapide, le disque en général (cf. les appareils virtuels du chapitre 4): quand l'application croira écrire sur l'imprimante, elle écrira en fait dans un fichier disque (fichier "SYSOUT"). L'imprimante qui est par nature un appareil non partageable pourra ainsi être utilisée de façon partagée: plusieurs processus écriront simultanément sur l'imprimante. Ils écriront en fait dans des fichiers disque différents.



Un composant du SE, généralement externe au noyau, assurera le vidage de ces fichiers "SYSOUT" sur les imprimantes réelles. On parlera donc d'impression **différée** (differed print, outspooling dans RTE-A), réalisée par les "output writers" de JES (autrefois inclus dans l'OS), les "daemon" de Multics ou GCOS6, les "symbionts" de l'OS1100 ou des SIRIS de l'ex-CII, les **spouleurs** plus généralement, terme venant du "spooler" américain, "spool" étant un acronyme pour "Simultaneous Peripheral Operation On-Line" que l'on trouve dès les années 60 comme option dans le SE de l'IBM 1401. Dans le cadre de ce tome nous utiliserons le terme de Gestionnaire des Périphériques lents" (GPL).

Ce que nous venons de décrire dans le cas de l'impression a son pendant côté lecture avec la gestion du SYSIN déjà introduite. Cette fonction a été ajoutée dans MS-DOS version 2.10 sous forme de commande PRINT qui permet d'imprimer un fichier pendant que l'utilisateur déroule une autre activité sur son écran/clavier. Dans les gros et moyens systèmes une commande telle que le PRINT de MS-DOS conduit généralement à une recopie du fichier à imprimer dans le "spool" ou SYSOUT et l'utilisateur peut sans problème supprimer son fichier dès que le SE a pris en compte la commande PRINT. Il n'en va pas toujours ainsi dans les systèmes MS-DOS à disquette: la suppression d'un fichier soumis à la commande PRINT rendra impossible l'impression complète de ce fichier.

Figure 6p: configuration "delta".



6.5.2 – Cas de processeurs spécialisés.

Certains systèmes ont été conçus pour gérer les périphériques lents de façon optimale, c'est-à-dire en les éloignant du processeur déroulant les applications. Ainsi dès 1962 on a vu des ordinateurs mis en frontal devant l'ordinateur "principal": c'est la cas du DCS (Direct Coupled System) d'IBM dans lequel un ordinateur 704X gérait les périphériques lents ainsi que les lignes de communication d'un 709X (figure 6q), les deux ordinateurs étant connectés par une liaison reliant deux de leur canaux (liaison "canal à canal" ou CTC pour Channel To Channel). On trouve aussi ce genre de configuration dans cer-

tains systèmes 1100 de Sperry, dans la configuration ASP (Asymetric Support Processor) sur la série 360 d'IBM en 1968. Pour les terminaux lourds sur GCOS3, le frontal jouait en partie le rôle de GPL dans les configurations dites "delta" (figure 6q).

Il ne faut pas confondre ce type de configuration avec les contrôleurs de périphériques que l'on trouve sur les canaux des gros et moyens ordinateurs. Ainsi l'URC (Unit Record Controller de la série 64 de Bull, devenu le SURP sur les DPS7 actuels) gère lui aussi les lecteurs de carte, les imprimantes, les lignes de communication, mais il ne les gère qu'au niveau physique, il ne joue pas le rôle de GPL, il n'est responsable ni du SYSIN ni du SYSOUT.

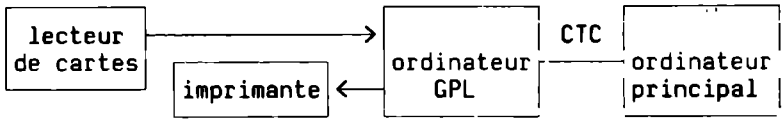


Figure 6q: configuration du type DCS ou ASP.

6.6 – Entrées/sorties virtuelles.

Nous verrons plus loin que nombre de systèmes d'exploitation gèrent leur mémoire centrale avec des techniques liées à la notion de mémoire virtuelle. Le gestionnaire de la mémoire centrale est alors doté d'un mécanisme lui permettant de vider temporairement sur disque des blocs (des pages, des segments) puis de les ramener en mémoire lorsqu'ils y sont nécessaires et que la place y est disponible. Les entrées/sorties associées à ces mouvements de blocs seront appelées des E/S virtuelles et des systèmes comme MVS permettent de les utiliser pour stocker des fichiers temporaires, c'est-à-dire des fichiers qui n'existent que pendant le déroulement d'un travail (job): il suffira pour cela de mettre la valeur VIO dans le paramètre UNIT de la commande // DD, une partie de l'AS est utilisée comme tampon. Il y aura gain de temps: pas d'allocation de périphérique, la gestion des données est moins lourde, l'espace disque est mieux utilisé, la taille disque est allouée de façon dynamique, et dans le cas de MVS on bénéficie du dispositif d'équilibrage des E/S (I/O balancing): le gérant d'EXCP intercepte cette E/S et la passe à VIO (ASM, figure 6s) au lieu de la donner à l'IOS (GESP); MVS parle alors de "fenêtre" (window) au lieu de tampon.

TAILLE AM EN Ko	0	16	32	64	128	256	384	512
COPIE NUMERO 1	7,4	7,4	4,7	5,4	5,6	5,2	5,3	5,3
COPIE NUMERO 2	7,6	7,6	4,6	4,7	5,8*	2,7	1,2	0,9

* résultat non expliqué

Tableau 6r.

Sans que l'on puisse parler de mémoire virtuelle dans MS-DOS versions 2 ou 3, des utilitaires comme RAMDISK ou MEMBRAIN permettent aussi de simuler en mémoire centrale des fichiers, et même des unités de disquette; de même l'option DEVICE=VDISK dans CONFIG.SYS. [UNG86] illustre les effets d'une "antémémoire" créée pour les disques par

l'utilitaire FXP.COM en copiant plusieurs fois de suite un même fichier de 80 Ko; le tableau 6r donne les temps d'exécution des deux premières copies.

Dans VMS le terme d'E/S virtuelle a un autre sens, il est associé à la notion de "bloc virtuel". Un bloc virtuel a une adresse virtuelle de bloc, de 1 à n, à l'intérieur de son fichier. Au niveau du volume, ce bloc a une adresse logique de zéro à n. Ce bloc (512 octets dans VMS) correspond à 1 ou plusieurs blocs physiques sur le volume selon que les secteurs y font 512, 256 ou 128 octets. Cet exemple de VMS illustre l'utilisation fréquente du terme "virtuel" pour représenter un second niveau de notion logique.

La gestion des entrées/sorties physiques sous MVS fait apparaître quatre éléments principaux: l'IOS lui-même, des gérants (drivers) de requêtes jouant un rôle d'intermédiaire entre les méthodes d'accès et l'IOS, les utilisateurs eux-mêmes, comme l'illustre la figure 6s tirée de [IBM80C]. Les méthodes d'accès utilisent le gérant EXCP qui portent le nom des macro-instructions EXCP et EXCPVR qui permettant de l'appeler.

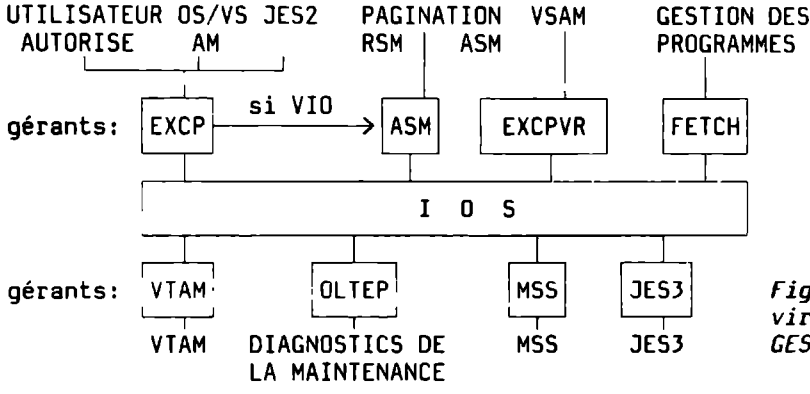


Figure 6s: l'environnement du GESP de MVS.

Le gérant EXCP est composé de trois parties:

- la partie frontale (front end) chargée de la traduction d'adresses, de la fixation des pages et de la construction des blocs de contrôle pour l'IOS. Les pages doivent être fixées car dans les 370 les canaux ne connaissent généralement que les adresses réelles et non pas les adresses virtuelles;
- la partie DIE (Disabled Interruption Exit) qui permet aux programmes utilisateurs d'être avertis lorsqu'une des commandes d'entrée/sortie du programme canal (cf. tome III) est terminée. Ceci est appelé PCI (Program Controlled Interruption) par IBM et correspond aux interruptions intermédiaires. Le programme peut ainsi modifier le programme canal au cours de son exécution;
- la partie d'arrière plan (back end) chargée d'analyser l'état de la fin de l'entrée/sortie. Elle se déroule sous contrôle du SRB et réactive (instruction POST) le processus utilisateur;

comme le montre la figure 6n. L'IOS a de même trois parties

principales:
- le distributeur de canal (canal scheduler) qui lance les entrées-sorties par l'instruction machine SIO (Start I/O) s'il existe un chemin disponible vers le périphérique concerné, (après avoir rempli la zone mémoire CAW utilisée par le canal comme une sorte de boîte aux lettres) ou bien met en file d'attente (queue) la demande en cas de non disponibilité d'un chemin. C'est lui aussi qui va chercher dans ces mêmes files d'attente, la prochaine requête (programme canal) à lancer lorsqu'une interruption de fin d'E/S libère un chemin;
- le gestionnaire des interruptions (dit de second niveau) appelé par le premier niveau à la suite d'une interruption générée par un canal (voir tome III). Il regarde si une E/S en attente peut être activée. Il lance aussi une tâche SRB chargée d'assurer le traitement complet de la fin de l'entrée/sortie, redonne la main au programme interrompu ou bien au distributeur (du processeur central);
- la routine de traitement final de l'état de l'E/S (post status) exécutée par la tâche SRB lancée par le gestionnaire d'interruption.

Les évolutions d'architecture apportées par les 308X (XA) ont déporté la majeure partie de l'IOS du processeur central vers le processeur d'entrée/sortie (appelé EXDC). Ce point sera développé dans le tome III.

6.7 – Cas des multiprocesseurs.

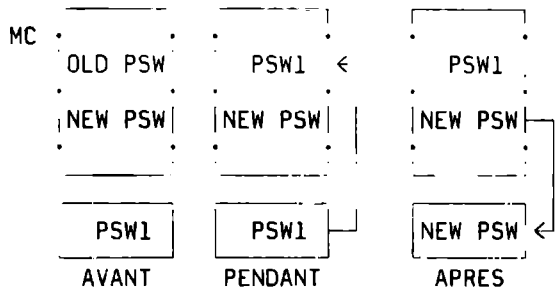
Comme nous l'avons souligné dans le tome I, selon le degré de banalisation des ressources, chaque processeur pourra être limité à un sous-ensemble des canaux, sinon des périphériques. C'est le cas des 370 avant XA, et lorsqu'un processeur lance une E/S sur un périphérique, si aucun chemin physique n'est disponible, il remet en file d'attente l'E/S (requeue), exécute une instruction SIGP (SIGnal Processor) qui interrompt l'autre processeur. Le distributeur prend la main dans cet autre processeur et prend en charge l'E/S; un indicateur précise que l'E/S vient de l'autre processeur afin d'éviter les parties de ping-pong ([DUB86]).

6.8 – Gestion des interruptions (GINT).

Sous MS-DOS la gestion des interruptions est réalisée principalement par le BIOS et le BDOS (interruption 21H). La nature vectorisée des interruptions et l'utilisation de la pile en simplifient la gestion, il n'y a cependant qu'un seul niveau de priorité d'interruption, comme dans le cas d'Unix et de MVS ou VSE, pour ce qui concerne les entrées/sorties.

Sous MVS c'est le rôle des routines IH (Interrupt Handler) dont il existe six types:
- celles traitant les SVC (appels au superviseur). Il faut alors déterminer où se trouve la routine de cet SVC (dans le noyau ou en LPA), si l'appelant a droit à cet SVC (programmes dits "autorisés"). On distingue deux sous-types de SVC selon qu'elles peuvent ou non appeler une autre routine SVC. Une routine pouvant à son tour provo-

quer l'exécution d'une autre SVC (devenir interruptible) doit construire un bloc de contrôle SVRB pour sauvegarder son contexte: il n'y a en effet pas de pile sur les 360/370 et successeurs, mais un simple dispositif d'ancien (old) et nouveau (new) mot d'état (PSW). Quand une interruption arrive le mot d'état courant du processeur (current PSW) est sauvegardé dans une zone mémoire associée au type d'interruption et appelée "old PSW". Le contenu d'une seconde zone mémoire associée à cette interruption, appelée "new PSW", est alors chargé dans le mot d'état du processeur et devient le mot d'état courant: il contient l'adresse de la routine de traitement de cette interruption. Une convention liée au système d'exploitation, et non pas au matériel, permet de sauvegarder les registres dans une zone de la première page (4 Ko) de la mémoire centrale;



Sauvegarde du mot d'état (PSW) et chargement de l'adresse du programme de traitement de l'interruption apparue (NEW PSW) sur 370. Il y a six couples [OLD,NEW] PSW pour les six types d'interruption.

INTERRUPTION

- celles associées aux **entrées/sorties** qui provoquent l'activation du GESP (IOS) et constituent ce qu'IBM appelle le gestionnaire d'interruption de premier niveau (first level IH), la partie gestion des interruptions de l'IOS constituant le second niveau;
- les interruptions de **programme** qui appellent:
  * RSM en cas de défaut de page (cf. chapitre 11),
  * GTF si l'interruption est due à un événement de contrôle (monitor event),
  * un programme utilisateur de traitement de l'interruption (créé par la macro SPIE, Set Program Interrupt Element),
  * RTM (Recovery Termination Manager) en cas d'erreur dans le code du système lui-même ou bien si l'utilisateur n'a pas de programme de traitement de l'interruption,
  * SLIP (Serviceability Level Indication Processing) si l'interruption est associée à un contrôle de chargement (fetch), de branchement réussi (successful branch), de modification d'une zone mémoire, arrêt sur adresse;
- l'interruption de **redémarrage** (restart) qui provoque l'appel de RTM responsable des reprises dans MVS;
- l'interruption d'**incident machine** (machine check) qui conduit aussi à l'activation de RTM;
- les interruptions **externes** (external) qui concernent les horloges, le dialogue entre processeurs.

chapitre 7

gestion des fichiers

7.1 – Les différents niveaux.

Qui dit gestion des données évoque l'accès à des informations stockées sur des périphériques, ce qui suppose des relations avec la gestion des périphériques, les volumes qui y sont montés (cf. tome IV), et les fichiers qu'ils contiennent. Dans ce chapitre nous traiterons principalement de la gestion des fichiers, leurs relations avec celle des volumes et des périphériques seront précisées en fin de chapitre, étant entendu que certains périphériques ne sont pas utilisés pour stocker des données, mais seulement pour communiquer avec l'extérieur, comme les écrans/claviers, les imprimantes, etc.

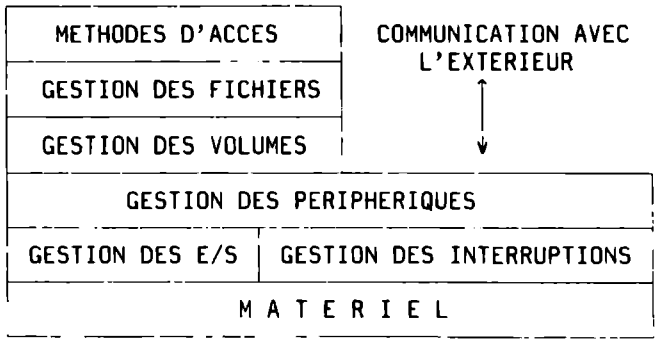


Figure 7a: hiérarchie des notions associées à la gestion des données.

Dans un système informatique, la gestion des données se traduit par:
- la présence de plusieurs niveaux de gestionnaires, comme le montre le modèle en couches de la figure 7a;
- le double aspect du **contenant** (le fichier, la base de données) et du **contenu** (les articles).

Si les méthodes d'accès sont extérieures au noyau du SE, la gestion des fichiers fait par contre généralement partie de ce noyau.

Le système de gestion des fichiers (SGF) est accessible aussi bien par le langage de commande du SE (le JCL) que par une interface

programmatique. Dans le cas des OS d'IBM la commande // DD permettra d'accéder aux services du SGF, pour GCOS3 ce seront les commandes $ FILE ou $ PRMFL.

7.2 – Les fonctions du SGF.

Le système de gestion des fichiers gère le contenant des informations traitées par les utilisateurs en assurant les fonctions listées ci-dessous:

- le **répertoire** du SGF (directory, catalog) permet aux utilisateurs de manipuler les fichiers par **nom** et non pas par adresse. L'utilisateur n'a donc pas à se soucier du volume sur lequel réside tel fichier, ni à quel endroit exact sur le volume le fichier est stocké. Toute liberté est ainsi laissée au SE qui choisit en fonction de l'état des ressources disponibles, où ranger tel fichier;

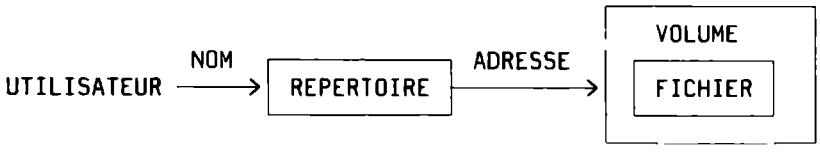


Figure 7b: la fonction répertoire.

- la **protection** de l'accès au fichier par des mécanismes de droits d'accès mémorisés dans le répertoire;

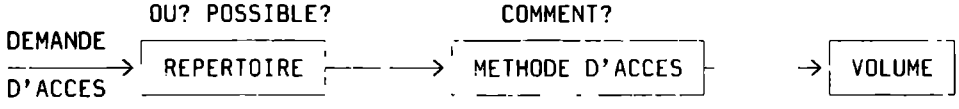


Figure 7c: rôles respectifs et successifs du répertoire et des méthodes d'accès.

- la journalisation des demandes d'accès dans un souci analogue de contrôle, de sécurité, c'est-à-dire constitution d'une trace (audit) de tous les accès;
- le contrôle de l'**accès concurrent**, situation créée par plusieurs utilisateurs qui veulent manipuler simultanément le même fichier. Selon les systèmes il y aura contrôle d'accès concurrent au niveau fichier, au niveau piste, au niveau bloc, au niveau article et donc implication de la méthode d'accès dans certains cas;
- la gestion automatique des **versions** (terminologie VMS, génération chez IBM) d'un même fichier;
- une **sécurité** accrue contre la perte ou la détérioration du fichier: gestion d'une seconde copie du fichier, journalisation des mises à jour, mises à jour différées, mode test qui fait que les modifications ne sont pas réellement reportées dans le fichier (GCOS3), répertoire dupliqué (VSAM), des outils de reconstruction des fichiers détériorés. RMS de VMS offre la possibilité de lire deux fois puis de comparer (mise en oeuvre par la commande FDL FILE READ_CHECK); dans le cas de l'écriture d'un fichier, la com-

mande FILE WRITE_CHECK déclenche une lecture après écriture, puis la comparaison. Dans ces deux cas le coût est important puisque les entrées/sorties sont doublées;
- des outils de manipulation de fichiers: copie, comparaison, changement de nom, affichage, vidage (dump), liste du répertoire, inscription au répertoire;
- gestion des **statistiques** d'utilisation, de place disponible;
- gestion de **synonymes** (alias) permettant de donner plusieurs noms à un même fichier;
- assimilation des **périphériques** à des fichiers afin de banaliser au maximum les ressources périphériques.

[GIE85] mentionne que les trois quarts des fonctionnalités du noyau d'Unix concernent la gestion des fichiers.

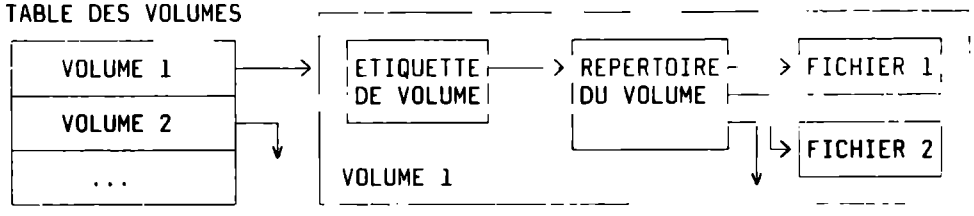


Figure 7d: volume contenant son propre répertoire.

7.3 – Structure du répertoire.

Dans le cas le plus simple, le SGF gère un répertoire **plat**, analogue à un tableau, avec une ligne (on dira aussi une entrée) par fichier inscrit au répertoire: les fichiers qui y sont inscrits seront dits **catalogués** (cataloged) puisque répertoire se dit "catalog" (ou catalogue) en américain. On devrait dire "répertorié" mais ce qualificatif n'est pas d'usage courant. Le paramètre DISP=(,CATLG) de la commande // DD de MVS permet de définir un fichier catalogué c'est-à-dire de l'inscrire dans le catalogue.

Dans d'autres SGF le répertoire a une structure plus **décentralisée** et chaque volume contient une partie du répertoire, celui concernant les fichiers qui sont stockés sur le volume. Le SE dispose d'une table des volumes en ligne (on line) et à un endroit convenu pour le SE en question se trouve le répertoire du volume, ou un pointeur vers ce répertoire. Ainsi dans GCOS3 le SMC (System Master Catalog) se trouve à un emplacement défini par le onzième mot de l'étiquette (label) standard du volume. Une étiquette de volume décrit le volume (cf. tome III). MS-DOS fournit un exemple de ce type de répertoire (tableau 7e), répertoire placé à un endroit convenu, après la table d'allocation des secteurs; l'entrée décrivant le volume peut par contre se trouver n'importe où dans le répertoire, son attribut de type (L) l'identifie et dans l'exemple du tableau 7e elle figure dans la quatrième entrée.

7.3.1 – Sous-répertoires.

Les structures plates, centralisées ou décentralisées au niveau d'un volume, posent un problème du fait du grand nombre d'entrées (de fichiers) qu'elles peuvent être amenées à contenir. Afin de rendre plus maniables ces répertoires, afin aussi d'offrir plusieurs environnements, de nombreux systèmes se sont dans la foulée de Multics dotés de répertoires **arborescents**. MS-DOS en est un exemple depuis la version 2, un sous-répertoire dans le répertoire d'une disquette étant identifié par une entrée de type 10 (au sens de la figure 7e) en hexadécimal.

L'ensemble des volumes figure dans un répertoire fictif appelé **racine** (root) et constitue le **niveau** le plus élevé de l'arborescence, de la hiérarchie. Chaque volume contient à son tour un répertoire arborescent avec des sous-répertoires comme le montre la figure 7f. L'ensemble constitue un graphe dont les noeuds sont les répertoires et sous-répertoires et les feuilles les fichiers.

Le **nom** d'un fichier est formé de plusieurs éléments qui décrivent le **chemin** qui conduit de la racine au fichier. Ainsi le nom **complet** (ou absolu) du fichier FICHIERij4 de la figure 71 s'écrira:

VOLUMEi:[REPERTOIREij]FICHIERij4

sur VMS. Dans le cas de MS-DOS un tel nom s'écrira:

VOLUMEi:\REPERTOIRE\FICHIERij4

par exemple, le nom de volume représenté ici par VOLUMEi étant bien évidemment réduit à une seule lettre.

```
┌──────────────────────────────────────────────────────────────────────┐
│        NOM          SUFFIXE  TAILLE   DATE   TYPE ADRESSE             │
│1: 494F202020202020 535953 BC2E0000 3477870B 27   0200                 │
│2: 4D53444F53202020 535953 7B420000 9C002F08 27   0E00                 │
│3: 434F4D4D414E4420 434F4D F53E0000 366B3107 20   1F00                 │
│4: 53595354454D2D4C 4F5455 00000000 A3012100 08   0000                 │
│ ^                                                                     │
│ │──────── numéro de l'entrée dans le répertoire                       │
│ v                                                                     │
│1: IO                SYS     11964o  07/12/85 HSRA  2 ┌──┐             │
│                                       14H57           │TYPE:           │
│2: MSDOS             SYS     17019o  15/01/85 HSRA 14 │L: volume label │
│                                       12H04           │N: normal       │
│3: COMMAND           COM     16117o  17/09/83 NA   31 │H: hidden(caché)│
│                                       13H25           │S: system       │
│4: SYSTEM-LOTU                       01/01/80 L        │R: read-only    │
│                                       12H13           │A: archive      │
└──────────────────────────────────────────────────────└────────────────┘
```

Figure 7e: contenu d'un répertoire MS-DOS.
(en hexadécimal en haut, interprété en bas)

7.3.2 – Répertoire courant.

La majorité des SE dotés d'un tel répertoire arborescent offre la notion de répertoire courant (working, current directory) qui permet à l'utilisateur de manipuler des noms locaux, plus courts que les noms complets, donc plus rapides à écrire. Un autre avantage de ces noms locaux est qu'ils permettent d'accéder à des fichiers différents s'ils sont utilisés dans des répertoires différents. Ainsi le résultat de la commande TYPE NOMS sur un système MS-DOS dépendra du sous-répertoire dans lequel on la frappe dans la figure 7g. Le SE offre pour se déplacer dans l'arborescence un jeu de commandes et de primitives: CD (Change Directory) dans le cas de MS-DOS, SET DEFAULT sur VMS, WD sur RTE-A, etc.

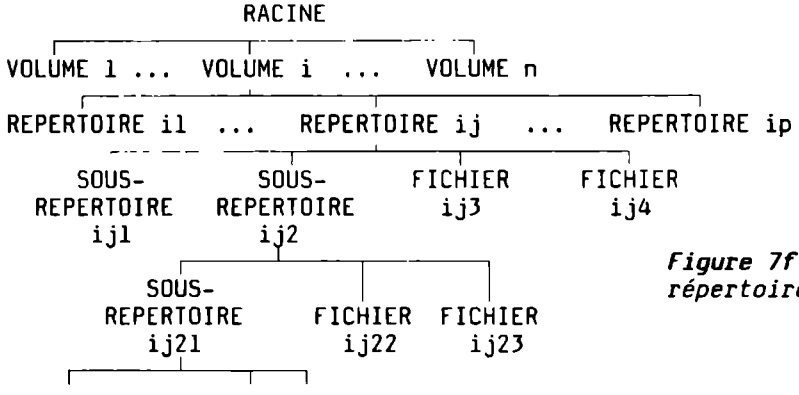


Figure 7f: exemple de répertoire arborescent.

Dans un système MVS, la racine correspond au répertoire maître (master catalog) aussi appelé "système" (system catalog). Ce nom est dû au fait que ce répertoire est nécessaire au fonctionnement du système. Il localise les autres répertoires et les fichiers répertoriés (catalogués) dont il contient le nom et le type du volume (serial number). A la différence des SE MVT et SVS, ce répertoire maître est maintenant en VSAM sous MVS (voir plus loin les SE multi-SGF).

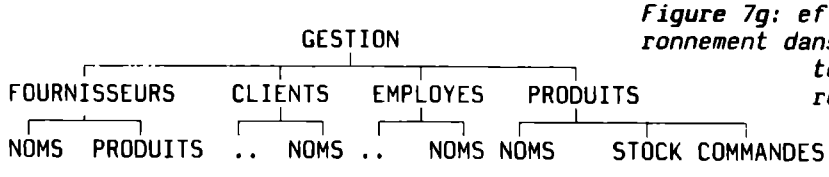


Figure 7g: effet d'environnement dans les répertoires arborescents.

7.3.3 – Relation utilisateur–répertoire.

Il n'y a, a priori, aucune raison de lier un utilisateur à un répertoire ou sous-répertoire particulier, chaque utilisateur pouvant selon ses droits d'accès se positionner à l'intérieur de l'arborescence globale. Quand l'utilisateur se connecte (login) il y a

cependant un positionnement par défaut effectué par le système; dans certains SE un des niveaux de répertoire doit avoir le même nom que l'utilisateur (Primos, GCOS3). dans d'autres SE un des attributs associé à l'utilisateur définit le répertoire de connexion (SYS$LOGIN de VMS).

7.3.4 – Fichier multirépertoire.

La plupart des SE offrant des répertoires arborescents imposent à chaque fichier de ne figurer que dans un seul répertoire ou sous-répertoire: il n'existe qu'un seul **chemin** allant de la racine à un fichier. Quelques SE permettent aux fichiers d'être inscrits dans plusieurs répertoires comme le montre la figure 7h. Un fichier y aura alors plusieurs **noms**, les différents chemins qui y mènent imposeront différents droits d'accès. VMS, Unix et Multics offrent cette possibilité de fichier multirépertoire.

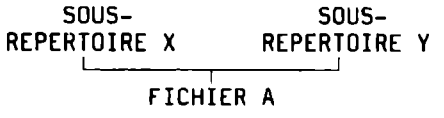


Figure 7h: cas d'un fichier appartenant à plusieurs répertoires.

7.3.5 – Extension au réseau.

Dans le cas de machines reliées par un réseau relevant de l'architecture réseau d'un constructeur (cf. chapitre 13) on pourra assister à une généralisation de cette notion de répertoire à l'ensemble du réseau, un ordinateur y apparaissant sous son nom ajouté au nom local du fichier (cf. figure 7i). Dans le cas de VMS, le fichier FICHIERIij4 supposé stocké dans la machine x aura le nom suivant:

MACHINEx::VOLUMEXi:[REPERTOIREij]FICHIERij4

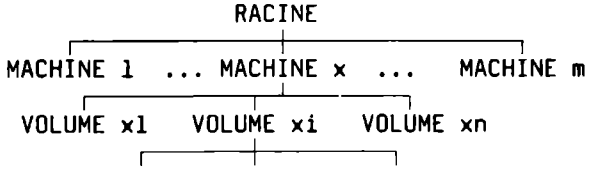


Figure 7i: répertoire étendu au réseau.

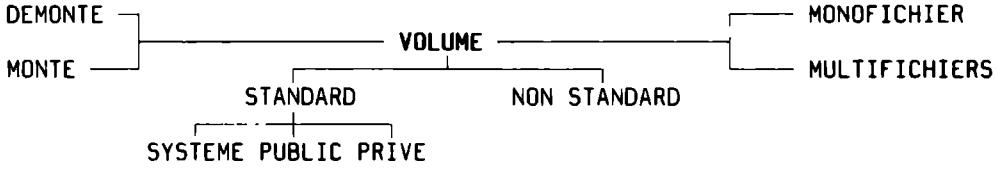
7.4 – Types de volume et de fichier, relations.

La notion de volume est une notion que nous avons présentée comme étant de nature **physique**. Elle correspond à un objet que l'on peut manipuler, déplacer comme un tout: une pile de disques, une bande magnétique, une cassette, une disquette, etc. Dans le cadre du SGF, le volume est un objet qui est soit en ligne, soit hors ligne, accessible, ou non accessible. Un volume peut très bien être "monté" dans une unité de disque et ne pas être logiquement connu du SE, à la suite d'une commande d'exclusion (démontage) du pupitreur par exemple. Dans le contexte d'un réseau d'ordinateurs, un ordinateur peut très bien ne pas "mettre" sur le réseau tous ses volumes.

Ces notions de "en ligne" et "d'accessibilité" conduisent la plupart des systèmes à implémenter la notion de volume **logique**. Dans certains cas (Primos, RTE-A, Unix) le volume logique est une partie d'un volume physique, dans d'autres cas (Multics) il comprend plusieurs disques physiques. Le "volume set" (ensemble de volumes) de VMS correspond aussi à plusieurs volumes physiques vus comme un seul volume logique par les utilisateurs.

Un volume dit **système** contient le système d'exploitation et certains de ses fichiers, il est obligatoirement en ligne dans la grande majorité des SE. Un volume **public** est un volume accessible a priori par tous les utilisateurs. Un volume **privé** n'est en ligne que lorsque son propriétaire en demande le montage.

Certains SE gèrent le prémontage des volumes physiques: dès qu'un périphérique passe dans l'état prêt, une attention (interruption asynchrone, cf. tome III) est émise vers le sous-système central qui va ensuite lancer les entrées/sorties nécessaires pour reconnaître le volume monté grâce à son étiquette (label) de volume. Cette fonctionnalité est parfois abrégée en AVR (Automatic Volume Recognition) et permet, entre autres, de ne pas prendre un volume pour un autre. Il n'en va pas ainsi par exemple sous MS-DOS qui est totalement aveugle au changement de disquette, et qui, pour se prémunir contre tout changement de disquette, relit le répertoire à chaque commande DIR !



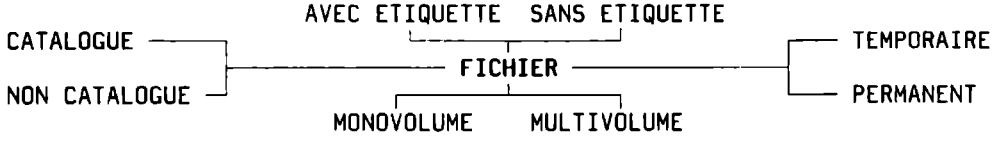
Un volume **structuré** ou **standard** correspond généralement à un volume dont la structure est compréhensible par le système d'exploitation: l'étiquette de volume se trouve à l'endroit convenu avec la structure convenue, le répertoire a une structure reconnue par le SGF, etc. Les disquettes "autochargeables" sur PC sont fréquemment non standard vis-à-vis de MS-DOS qui est incapable d'y retrouver un répertoire. La terminologie informatique étant ce que nous savons, iRMX86 utilise le terme "structuré" pour les fichiers sur disque ou en mémoire centrale, Enscribe de Guardian l'utilise pour les fichiers avec une organisation séquentielle, indexée ou relative, par opposition aux fichiers binaires contenant du code généré par exemple.

Un volume **étiquetté** (labeled) contient une étiquette de volume qui identifie le volume. Les bandes n'ont pas toujours ce type d'étiquette (bande NL, NoLabel chez IBM) et lorqu'elles en disposent cette étiquette peut correspondre à un standard du SE (SL comme Standard Label chez IBM), à un standard international (ANSI) ou national (AFNOR). Cette étiquette de volume joue aussi un rôle de sécurité dans la mesure où elle permet au SE de vérifier que le volume monté est bien celui demandé: un opérateur peut se tromper dans le montage d'une bande, la bande peut être mal étiquettée à l'extérieur. L'étiquette

magnétique peut de plus contenir un mot de passe que l'utilisateur devra fournir.
MVS définit deux attributs pour les volumes disque (cf. [JES80C]):
- l'attribut de **montage** qui peut avoir la valeur PRES (résident de façon permanente, cas des volumes contenant SYS1.LINKLIB, les fichiers de pagination par exemple), REMV (REMoVable, démontable), PERManent;
- l'attribut **d'utilisation** qui peut prendre les valeurs STR (stockage), PUB (public), PRIV (privé, paramètre VOLUME=PRIVATE dans la commande // DD).

Tout comme il y a des articles multibloc (format "spanned") et des blocs multi-article, il existe des **fichiers multivolumes** et des volumes multifichiers. Les disques sont généralement multifichiers, les fichiers multidisques ne sont, par contre, pas toujours supportés. De façon analogue aux volumes, les fichiers peuvent avoir une étiquette (file label) ou en être dépourvus. Cette étiquette peut participer au contrôle de sécurité dans la mesure où des mots de passe y sont inscrits.

Un fichier **temporaire** est un fichier qui n'existe que pendant l'exécution d'une ou plusieurs étapes d'un même travail et est automatiquement détruit par le SE, au plus tard à la fin du travail. La notion de fichier temporaire est donc indépendante de celle de fichier catalogué, les deux notions se combinant si nécessaire. Unix les crée dans le répertoire "temp", MS-DOS les introduit dans la version 3.



7.5 – Gestion des périphériques.

La gestion des périphériques consiste à connaître en permanence leur état afin de réaliser les entrées/sorties demandées par les utilisateurs:
- en ligne ou hors ligne, c'est-à-dire accessible ou non;
- alloué à tel utilisateur ou disponible;
- le taux d'utilisation afin d'équilibrer les requêtes entre des périphériques analogues, afin de fournir des statistiques d'utilisation pour les futures opérations de facturation et de métrologie;
- le taux d'erreur afin de détecter une dégradation du fonctionnement;

Dans certains SE comme MVS (DDR de [JES80C]) il est possible de **reconfigurer** dynamiquement la périphérie en déplaçant un volume démontable du périphérique défectueux sur lequel il était monté, pour le remonter et le repositionner sur un autre périphérique, sans que l'application l'utilisant soit perturbée. Le DDR de JES3 fonctionne ainsi sur une commande SWAP du pupitreur ou sur une action du SE due à de trop nombreuses erreurs d'E/S.

Un autre aspect de la gestion des appareils périphériques (peripheral device) est leur **intégration** dans le système de gestion de

fichier. Nous avons vu plus haut comment les volumes apparaissaient dans les répertoires. Les périphériques bénéficient ainsi des mêmes mécanismes de protection, les interfaces programmatique et externe sont plus simples pour l'utilisateur. Sous MS-DOS la commande:

```
COPY A:COVER1.DOC B:UTIL.TXT
```

copie un fichier de la disquette A: sur la disquette B: en en modifiant le nom. La commande:

```
COPY A:COVER1.DOC CON:
```

copiera le même fichier sur "CON:" qui apparaît en lieu et place du fichier B:UTIL.TXT. Or "CON:" est l'écran. Dans certains SE, les périphériques figurent dans ces répertoires tout comme des fichiers: cas d'Unix (fichiers "spéciaux" du répertoire /dev), iRMX86 (fichiers "non structurés"), Tripos. Dans Guardian les accès à un fichier se font de façon analogue à un dialogue entre processus, d'où une grande généralité des programmes: remplacer un terminal par un programme le simulant ne modifie en rien l'application; il n'en va pas de même avec d'autres SE comme GCOS6 par exemple où on utilise des primitives (appels système) différentes selon que l'on dialogue avec un terminal ou avec un processus ([CHA86]). Dans Guardian les gérants d'appareil sont implémentés sous forme de processus et la communication interprocessus est gérée par le SGF, les processus y sont d'ailleurs appelés "fichiers logiques". Il en va de même dans MS-DOS, en particulier depuis la version 2.0 avec les fichiers standards ("handle" ou numéros analogues aux pointeurs d'Unix) encore que les périphériques n'y sont pas rangés dans un répertoire global qui n'existe pas: il n'y a pas de notion de racine dans MS-DOS et chaque volume y est autonome: la commande DIR *:doc.* n'est pas comprise par MS-DOS.

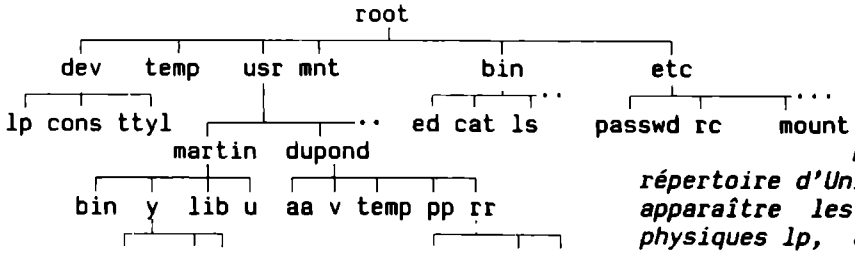


Figure 7j: répertoire d'Unix faisant apparaître les appareils physiques lp, cons, tty.

7.6 – Protection de l'accès.

Pour protéger l'accès aux fichiers il faut pouvoir identifier le demandeur et confronter sa demande avec les **droits** qu'il a sur l'objet. Un utilisateur a, au minimum, un nom, un numéro qu'il donne lorsqu'il accède au système. Le ou les processus qui vont agir pour lui (être ses agents) à l'intérieur du système informatique vont conserver ce nom ou porter une identification interne dérivée de ce nom (cas de l'UIC de VMS).

Les deux mécanismes de base utilisés pour protéger l'accès aux fichiers sont les **mots de passe** (password) et les **listes d'accès** (ACL, Access Control List). Si le demandeur produit le mot de passe nécessaire il pourra utiliser le fichier. Dans le cas des listes d'accès son identification interne lui donnera accès au fichier si elle se trouve dans la liste des utilisateurs agréés associée au fichier. Les deux mécanismes peuvent être utilisés simultanément (cas de VMS).

Lorsque l'accès est autorisé, le processus pourra réaliser sur le répertoire ou le fichier les opérations auxquelles il a droit:

- **création**;
- **lecture** (read): droit de lire le contenu du fichier, de lister le répertoire;
- **écriture** (write): droit d'écrire dans un fichier, de modifier un répertoire;
- **exécution** (execute): droit d'exécuter le programme contenu dans le fichier concerné. Le droit de lecture implique fréquemment le droit d'exécution;
- **suppression** (delete): droit de supprimer le fichier, le répertoire;
- **ajout** (append): droit d'écrire à la fin du fichier;
- **troncation** (truncate): droit de tronquer le fichier. On peut ainsi raccourcir le fichier, l'effacer, mais pas le supprimer;
- **modification** des caractéristiques, des protections.

Un objet (fichier, répertoire, volume) a un **propriétaire** et un créateur (originator) confondus ou différents selon les systèmes. Ainsi **Primos** qui utilise des mots de passe connaît deux types de demandeur: le propriétaire et le non propriétaire. Selon le mot de passe fourni, le demandeur se verra rangé dans l'une de ces deux catégories et aura droit aux opérations définies pour chacune d'elle: lecture, écriture, ajout, troncation. L'accès à des fichiers situés dans les couches basses des répertoires devient lourd si les mots de passe sont effectivement utilisés pour traverser les différents niveaux. L'écriture de fichiers de commande contenant les mots de passe pose des problèmes de sécurité.

Les listes d'accès, popularisées par **Multics**, offrent une solution d'autant plus souple que l'on peut grouper les utilisateurs; le nom d'un utilisateur a deux composantes: celle qu'il donne lors de la connexion plus un nom de projet. COMPTA.DUPONT sera le nom complet de l'utilisateur DUPONT travaillant sur le projet COMPTA. Le propriétaire d'un fichier pourra donner l'accès en lecture à tous les membres du projet COMPTA en nommant les utilisateurs COMPTA.*, l'astérisque représentant tous les membres de COMPTA. Syntaxe aisée à comprendre pour les utilisateurs de MS-DOS. Un fichier accessible à tous les utilisateurs aura *.* comme nom d'utilisateur agréé dans sa liste d'accès.

VMS utilise des listes d'accès, sur demande. Le mécanisme de base est une forme simplifiée de listes d'accès qui consiste à répartir les utilisateurs en quatre groupes. Il y a donc dans VMS une liste d'accès minimum de quatre entrées pour ces quatre groupes. DEC n'inclut cependant pas ce mécanisme dans ce qu'il appelle les listes d'accès. Les quatre groupes en question sont:

- le **propriétaire** de l'objet, c'est-à-dire celui qui a la même identification interne que son créateur (le même UIC, User identification Code);
- les membres du **groupe** du propriétaire, c'est-à-dire le propriétaire et ceux qui ont le même numéro de groupe que le propriétaire dans leur UIC;
- le **système**: les utilisateurs avec le privilège SYSPRV (SYStem PRiVilege), ceux dont le numéro de groupe (dans l'UIC) est inférieur au paramètre de génération MAXSYSGROUP, ceux du même groupe que l'utilisateur et avec le privilège GRPPRV (GRouP PRiVilege, des responsables de groupe en quelque sorte);
- le **monde** (world): tous les utilisateurs, y compris les précédents.

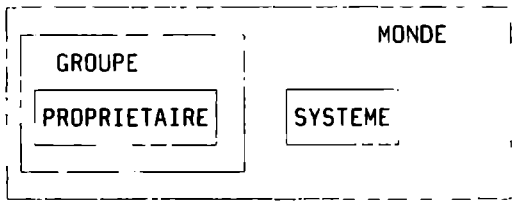
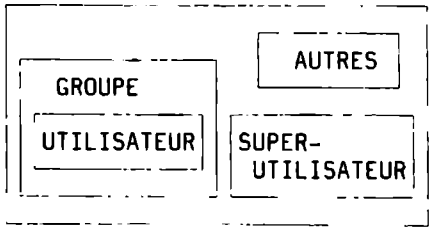


Figure 7k: catégories d'utilisateurs sur VAX/VMS.

Quand une requête est émise VMS regarde d'abord s'il existe une liste d'accès (ACL) pour l'objet (fichier, répertoire ou volume) et si l'ACL accorde l'accès. S'il n'y a pas d'ACL ou si l'ACL refuse l'accès à l'objet, VMS déclenche ensuite le mécanisme basé sur les UIC que nous venons de voir ci-dessus. Si ce deuxième mécanisme ne donne pas l'accès demandé, les privilèges sont ensuite analysés. Chaque entrée d'une liste d'accès de VMS permet de préciser s'il faut générer une alarme (impression sur un fichier journal par exemple) en cas de demande de lecture, d'écriture, d'exécution, de suppression, en cas d'accès réussi, en cas de refus.

Dans **iRMX86** la protection est basée sur un numéro d'utilisateur codé sur 16 bits, avec les opérations de lecture, écriture, modification (append), suppression. Chaque fichier peut en plus des droits de son propriétaire contenir une liste d'accès limitée à trois utilisateurs. L'utilisateur de numéro zéro est le superutilisateur (superuser) qui a tous les droits, le numéro 65535 représente un utilisateur quelconque, c'est-à-dire tout le monde (world).

Dans le cas d'**Unix** il y a aussi quatre types d'utilisateurs vis-à-vis d'un objet, mais trois modes seulement: lecture, écriture et exécution (recherche pour les répertoires).



Les catégories d'utilisateurs d'Unix: on notera la similitude globale avec VMS. AUTRES remplace MONDE mais ne correspond pas à la même définition: MONDE comprend tous les utilisateurs, AUTRES ne comprend pas GROUPE, ni UTILISATEUR, ni SUPERUTILISATEUR.

VM/CMS présente un cas un peu particulier dans la mesure où l'objectif initial est d'isoler totalement les différentes machines virtuelles, c'est-à-dire les utilisateurs. Un des moyens de communiquer entre utilisateurs est représenté par des volumes appelés **minidisques**. La commande LINK permet d'ajouter un minidisque à une machine virtuelle, de le rendre accessible à l'utilisateur de la machine correspondante:

LINK TO USERID ADR1 AS ADR2 PS PASS=mot-de-passe

Deux accès (PS ci-dessus) peuvent être demandés dans la commande LINK: un accès primaire "P", un accès secondaire "S" si l'accès primaire n'est pas possible.

MS-DOS a rendu visibles ces notions de protection avec la commande ATTRIB de la version 3: attributs RO (Read Only, lecture seulement), RW (Read/Write, lecture et écriture). CP/M 2.2. a de même introduit les attributs RO et RW.

VSAM offre 4 mots de passe, un par mode d'accès: plein accès (masterpw), accès CI (controlpw), accès en mise à jour (updatpw), accès en lecture (readpw). Une "exit" routine (chapitre 14) est activée quand le mot de passe du plein accès n'est pas donné ce qui permet à l'installation concernée d'ajouter ses propres contrôles.

7.7 – Accès concurrent.

Certains SGF offrent la possibilité de régler ou éviter les problèmes posés par l'accès simultané de plusieurs processus à un même fichier. Le SGF opère au niveau fichier, les méthodes d'accès opérant au niveau de l'article; selon le cas un seul ou bien les deux niveaux de contrôle seront offerts.

Au niveau fichier le propriétaire définira s'il peut y avoir plusieurs utilisateurs en lecture et/ou en écriture:

1L|1E: un seul utilisateur à la fois (mode exclusif), en lecture *ou* en écriture. Bloquer ainsi une ressource de taille importante peut créer de longues attentes pour d'autres processus;
nL|1E: plusieurs processus en lecture seulement *ou* un seul en écriture;
nL&1E: plusieurs "lecteurs" *et* un seul processus en écriture;
nL&nE: plusieurs processus en lecture ou écriture (ou inclusif).

Le SGF de **GCOS3** offre ainsi les modes "normal" (nL|1E), "RWW" (Read While Write ou nL&1E), "concurrent" (nL&nE contrôlés par l'utilisateur) et "moniteur" (nL&nE géré par le SGF).

Lorsque les méthodes d'accès ne gèrent pas automatiquement l'accès concurrent au niveau des articles, elles offrent parfois des primitives du type LOCK/UNLOCK (verrouillage/déverrouillage) pour permettre à l'utilisateur de régler lui-même le problème. Supposons que les deux processus P1 et P2 modifient le même fichier, P1 ajoute 2 au premier champ de l'article, P2 multiplie le second champ par 3. Si nous supposons de plus que les deux processus sont amenés à modifier l'article:

27	253

S'il n'y a aucun contrôle de concurrence au niveau des articles, l'article sera modifié différemment selon que P1 le modifie en premier ou en dernier et selon le temps écoulé entre ces deux modifications:

27	759

P1 en premier

29	253

P1 en dernier

29	759

pas de problème

Si l'on veut s'assurer qu'il n'y aura pas d'incohérence, P1 et P2 devront procéder de la façon suivante:

1. LIRE l'article et le verrouiller s'il n'est pas VERROUILLE
2. MODIFIER l'article
3. DEVERROUILLER l'article

Si l'article est verrouillé par un autre processus en 1., le processus demandeur est mis en attente et sera réveillé lorsque le possesseur fera l'opération de déverrouillage en 3.

Tous les systèmes ne verrouillent pas au niveau de l'article, certains bloquent la piste du disque contenant l'article, d'autres bloqueront le tampon, etc.

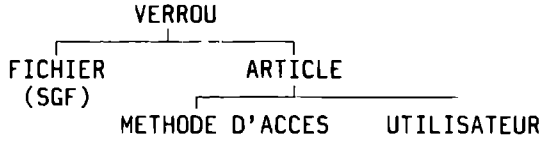


Figure 71: principaux niveaux de gestion des verrous.

Les attributs ACCESS et SHARING des commandes FDL de **VMS** permettent dans le cas du VAX de préciser le niveau de concurrence demandé:

- l'attribut ACCESS précise l'accès demandé: lecture (locate, read), écriture (delete, put, truncate, update), accès au niveau block (block_io, record_io);
- l'attribut SHARING précise la concurrence: accès exclusif, concurrence en lecture (nL), en écriture (nE), en écriture protégée (nLlE). Il permet encore de demander de gérer la concurrence par l'utilisateur, demander de lire (dans le même processus) plusieurs flots indépendants dans un même fichier (multistream).

La granularité du verrouillage y va du volume à l'item (champ) en passant par le fichier et l'article.

Dans le cas de **CMS**, et plus particulièrement de la commande LINK présentée ci-dessus, la demande d'accès au minidisque permet de demander:

- la lecture seule (nL): elle sera accordée si aucun autre utilisateur n'a le minidisque en écriture;
- lecture et écriture (accès exclusif): demande satisfaite si aucun autre utilisateur n'a le minidisque;
- multi-écriture (multiwrite): demande satisfaite même si un autre utilisateur a accès au minidisque en écriture.

Le minidisque dispose de trois mots de passe selon les modes d'accès possibles.

La commande SHARE de MS-DOS version 3 correspond à cette préoccupation d'accès simultané, dans un contexte de réseau local d'ordinateurs personnels puisque le PC reste mono-utilisateur dans cette version.

Sous MVS le paramètre DISP=SHR des commandes // DD permet de partager en lecture un fichier OS entre plusieurs travaux. Le paramètre SHAREOPTIONS(x,y) des fichiers VSAM permet de définir les modes nL|1E (x=1), nL&1E (x=2), nL&nE (x ou y égal à 3 ou 4).

7.8 – Cas des multisystèmes.

L'exemple de MS-DOS version 3 correspond non seulement au partage entre plusieurs utilisateurs, mais aussi au partage entre plusieurs ordinateurs qui ont chacun leur système d'exploitation, chacun leur SGF.

Le contrôle de l'accès concurrent peut être réglé brutalement en réservant au niveau appareil (device) seulement: un périphérique est attribué dans son entier à un seul système à un moment donné; les commandes RESERVE/RELEASE des OS d'IBM correspondent à ce type de contrôle d'accès concurrent.

Un exemple plus évolué, avec une **granularité** plus fine, est représenté par le mécanisme utilisé dans le VAXCluster où des sous-systèmes disque sont partagés entre plusieurs VAX, donc entre plusieurs copies de VMS. A chaque fichier est associé un VAX (appelé "noeud", "node") qui le contrôle à un moment donné et joue pour ce fichier le rôle de "maître du verrou" (lock master node). C'est initialement le premier noeud sur lequel est reçu une demande d'ouverture du fichier, mais ce noeud peut changer pendant le fonctionnement du système.

7.9 – Gestion des versions.

Le nom complet d'un fichier dans VMS a la structure suivante:

noeud::volume:[rép.srep.ssrep]nom.type;version

le numéro de version étant incrémenté chaque fois que le fichier est modifié. Une commande prend implicitement la dernière version du fichier et

DIFF FICHIER.FOR

comparera les deux plus récentes versions du fichier FICHIER.FOR. Afin de ne pas encombrer inutilement la surface disque il est possible de paramétrer le système pour qu'il ne garde que les n dernières versions des fichiers, couramment les deux ou trois dernières.

Dans d'autres systèmes, un nombre de volumes prédéfinis, des bandes généralement, forment une liste circulaire, chaque volume contenant une version du fichier.

7.10 – Domaines couverts par les SGF.

Les fonctions réalisées par les SGF varient fréquemment d'un SE à l'autre. Dans l'un ils ne géreront que le répertoire et l'allocation

de l'espace sur disque, dans un autre ils inclueront aussi les méthodes d'accès classiques: cas de VSAM chez IBM qui comprend toutes les fonctions présentées dans la figure 7m, mis à part les méthodes d'accès intégrées. Dans le cas de GCOS3 par contre, le SGF (FMS) ne s'occupera pas des méthodes d'accès, ce sera le rôle d'UFAS ou GFRC. VMS offre un cas analogue à GCOS3: le terme de FILE-11 correspond à la fonction de SGF ([VMS84C] parlera de "structure logique", de "disque FILE-11"), RMS correspond aux méthodes d'accès.

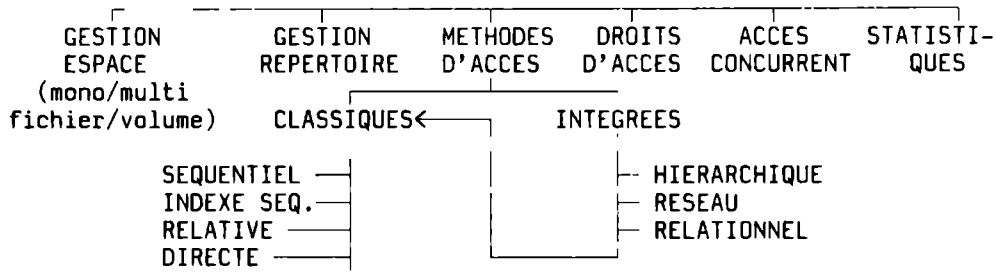
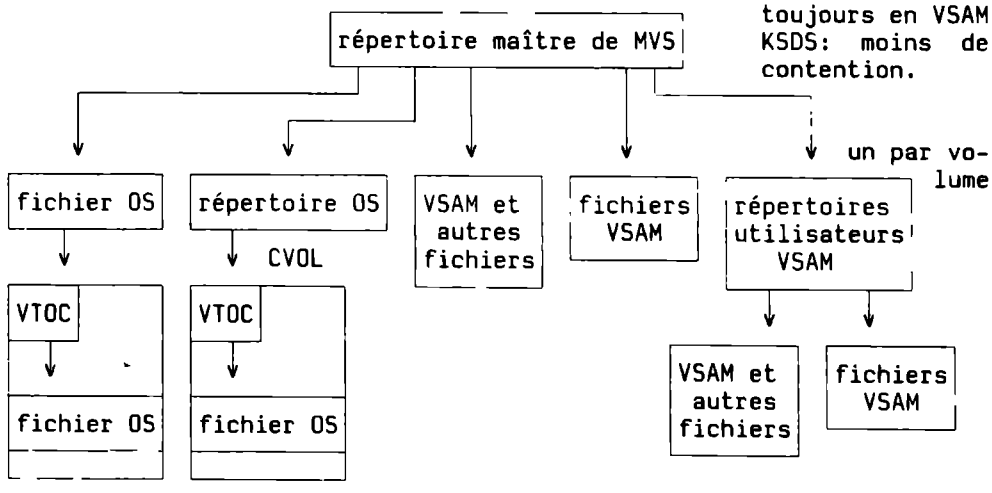


Figure 7m: les grandes fonctions d'un SGF.

7.11 – SE multi-SGF.

Un système comme GCOS3 comprend, comme nous l'avons vu plus haut, des méthodes d'accès "anciennes" (GFRC, ISP) initialement conçues pour des caractères 6 bits, des méthodes d'accès plus récentes comme UFAS, toutes deux définissent effectivement l'organisation du contenu des fichiers et sont compatibles avec le même gestionnaire de répertoire (appelé FMS).



Dans le cas de MVS il en va autrement car VSAM qui est de la même génération que UFAS comporte aussi la gestion du répertoire. Dans un

système MVS il y donc deux SGF, VSAM et l'ancien appelé "OS". Il y aura donc des fichiers OS et des fichiers VSAM, des répertoires VSAM, des répertoires utilisateurs OS (appelés CVOL). Un répertoire VSAM contient une entrée par volume (V) avec les caractéristiques du volume, les extensions attribuées aux espaces VSAM, les noms des objets VSAM, et 3 entrées par "cluster": une pour le "cluster" (C), une autre pour les données (D) et une dernière pour l'index (I). Un même volume contiendra des fichiers VSAM et des fichiers OS si besoin est. Dans la VTOC du volume apparaissent aussi les fichiers VSAM; seuls les espaces réservés à VSAM y étaient représentés précédemment, mais les problèmes de reprise (recovery) ont provoqué cette évolution.

7.12 – Taille des fichiers.

Selon [TAN83], dans un environnement Unix de chercheurs et d'étudiants, la taille des fichiers correspond à la distribution ci-dessous:

34 %	inférieurs à	512 octets	79 %	inférieurs à	4 Ko
52 %	"	1 Ko	88 %	"	8 Ko
67 %	"	2 Ko	94 %	"	16 Ko

Ces chiffres sont à rapprocher d'une enquête de [SCH80] sur un échantillon de 281 029 fichiers appartenant aux utilisateurs de 52 installations MVS: 65 à 70 % des fichiers ont une organisation séquentielle (SAM), la taille des blocs des fichiers SAM est inférieure à:

4 Ko	dans	85 %	des	cas
2 Ko	"	70 %	"	"
512 o	"	40 %	"	"

chapitre 8

gestion des données

L'évolution récente de l'informatique a mis en évidence l'importance primordiale des **données** stockées et gérées à l'aide des ordinateurs. Un système informatique **modélise** une partie de la réalité:

- par les données, les informations qu'il contient et qui décrivent cette réalité,
- par les programmes qui modélisent la façon dont les êtres humains traitent ces données.

Cette évolution et cette reconnaissance de l'importance des données est concrétisée par l'explosion des outils de gestion des données comme les **SGBD** (Systèmes de Gestion de Base de Données, DBMS pour Data Base Management System) qui suivent la norme CODASYL ou relèvent du type relationnel, par les **méthodes d'analyse** basées sur les données (data driven).

Une enquête d'IDC sur des machines installées aux Etats-Unis en fin 84 indiquait que 90 % des IBM 3084 avaient un SGBD; pour les 4381 ce taux passait à 70 %; 36 % des superminis avaient un SGBD, 16 % si l'on excluait les sites HP avec Image. Selon cette enquête le marché des SGBD devait être de 150 M$ en 1985, soit un accroissement de 35 à 40 % par rapport à 1984. Une autre enquête de Computer Intelligence Corp. portant sur 10000 sites équipés de 4300 d'IBM indique que 37 % des sites avaient un SGBD en 1982, 44 % en 1985.

8.1 – Les méthodes d'accès.

Nous avons souligné dans le chapitre précédent l'appartenance du SGF au SE et même à son noyau dans nombre de cas; les méthodes d'accès sont par contre extérieures au SE.

Elles correspondent à des interfaces programmatiques dans lesquelles on trouve des **primitives** (encore appelées macros, verbes) de lecture (READ), d'écriture (WRITE), de recherche (FIND), etc., d'articles. Le terme de méthode d'accès correspond à:

- du **logiciel**: les routines activées par ces primitives lorsque l'on désire manipuler des données stockées sous forme d'**articles** sur des

unités périphériques, routines rangées dans des librairies (bibliothèques, library) sur disque;
- des méthodes:
  * d'organisation de ces articles sur les supports,
  * d'accès à ces articles. La façon dont les données sont organisées permettra ou interdira telle ou telle méthode d'accès (cf. tableau 8k).

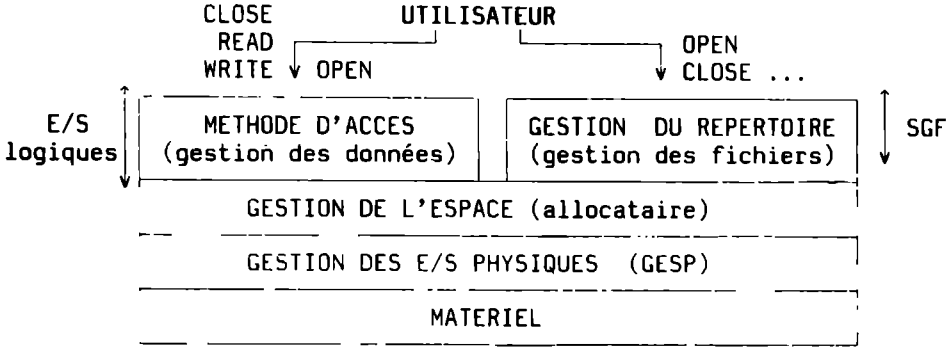


Figure 8a: les différents niveaux et aspects de la gestion des données.

L'organisation est une caractéristique **statique** d'un fichier ou d'une base de données. Elle est définie lorsque l'on **crée** ce fichier ou cette base de données. Le mode d'accès est une caractéristique **dynamique** choisie lors de l'**ouverture** du fichier ou de la base. Il pourra changer d'une ouverture à l'autre d'un même fichier.

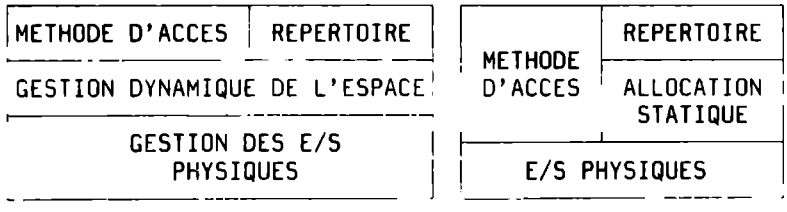


Figure 8b: comparaison entre gestion statique et dynamique de l'espace sur disque.

8.2 – Les types d'organisation.

Il convient de distinguer deux grands types d'organisation des données, celles associées aux fichiers et celles associées aux bases de données. Dans un langage de programmation comme Cobol 74, la clause ORGANIZATION IS définit l'organisation du fichier associé.

8.2.1 – Organisations des fichiers classiques.

Dans ces organisations, traitées de "standard" dans [IBM74X], les articles d'un même fichier sont **indépendants** les uns des autres, il n'y a aucune relation entre eux. Si relation il doit y avoir pour l'utilisateur, elle sera créée lors de l'exécution des applications

qui traitent ces articles, mais la façon dont leur stockage est organisé ne prend en compte aucune relation. Trois de ces organisations ont été normalisées èt reprises dans le Cobol 74 sous les noms de séquentiel, relatif et indexé séquentiel.

Dans l'organisation **séquentielle** les articles (enregistrements, record) sont stockés en fonction de leur création. Le n^e article stocké sera rangé derrière le $(n-1)^e$ et ainsi de suite. Pour lire le n^e article il faut lire auparavant les n-1 précédents. Cette organisation est supportée par la plupart des périphériques: disque, bande, disquette, cassette.

Dans le cas de VSAM l'utilisateur peut utiliser le RBA (Relative Byte Address) qui donne l'adresse du premier caractère de l'enregistrement pour accéder à l'information; ce mode d'accès se retrouve plus classiquement sur les mini-ordinateurs (Unix, Primos), sur les grosses machines de CDC.

Dans l'organisation **relative** chaque article a un numéro "relatif" dans le fichier et est stocké en fonction de ce numéro. Seuls les disques, les disquettes, les tambours supportent l'organisation relative. Les articles sont généralement de longueur fixe (cas de VSAM).

Dans le cas de l'organisation **indexée séquentielle** (indexed sequential) les articles sont rangés en fonction d'une clé qui fait partie du contenu de l'article. Le fichier comporte alors deux types de données:

- les articles eux-mêmes,
- des tables d'**index** qui indiquent, en fonction de la clé, l'adresse où est stocké l'article:

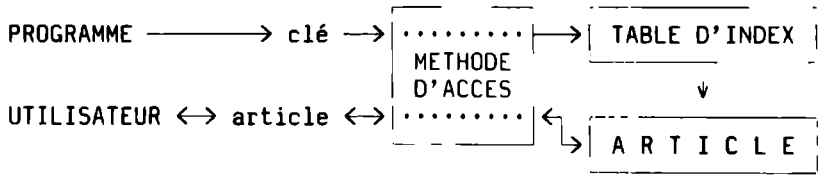


Figure 8c.

INDEX	ADRESSE
ALPH	4
BETB	5
CESI	6

Exemple de table d'index avec une clé de 4 caractères. Toutes les clés n'y figurent pas. Ainsi les clés dont la valeur serait comprise entre ALPI et BETA correspondent à des articles stockés dans le bloc numéro 4 ou à des articles inexistants.

Pour les gros fichiers la table d'index peut se révéler trop importante en taille et donc en temps de recherche. Elle fera à son tour l'objet d'une table d'index de niveau supérieur et l'on pourra arriver ainsi à n tables d'index disposées sous forme d'une structure hiérarchique à p niveaux, le niveau le plus élevé, donc de taille minimum, étant parfois appelé "index maître" (master index). Dans VSAM les

index de plus bas niveau sont appelés "sequence set", ceux des niveaux supérieurs "index set". RMS parle de niveau 1 (level 1) pour les index de plus bas niveau, de niveau 0 pour les enregistrements eux-mêmes.

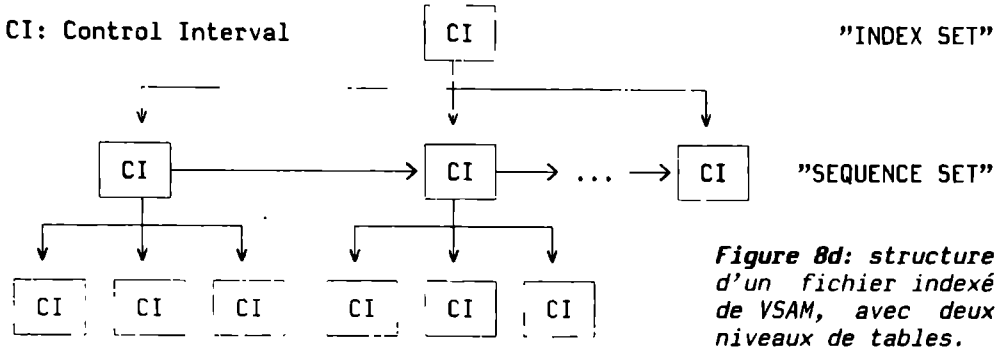


Figure 8d: structure d'un fichier indexé de VSAM, avec deux niveaux de tables.

La recherche d'un article provoque alors jusqu'à p+1 accès au disque, un accès par niveau d'index plus un accès pour l'article lui-même. Pour optimiser les performances certains niveaux sont stockés en mémoire: il y aura ainsi moins d'accès aux disques. Le tableau ci-dessous, tiré de [FRI81], montre les effets de plusieurs optimisations en utilisant deux types de disque, des 3350 et des 3380, en comparant deux méthodes d'accès indexées, ISAM et VSAM/KSDS. La lecture du fichier passe de 137 à 68 millisecondes avec un 3350 en ISAM, soit une division par deux, en VSAM le meilleur temps est de 22 millisecondes sur un 3380, soit près de sept fois moins qu'en ISAM sans optimisation sur un 3350.

ISAM	3350	3380
a TEMPS SANS OPTIMISATION	137,04	120,29
b INDEX MAITRE EN MEMOIRE	120,59	88,86
c INDEX SOUS LES TETES FIXES	116,35	97,06
b+c	99,90	80,39
d INDEX EN MEMOIRE	68,03	52,57
VSAM KSDS		
e CI de 16Ko/3350, 20Ko/3380	138,49	76,04
f CI de 2Ko/3350, 4Ko/3380	69,11	35,30
g +INDEX NIVEAU 2 SOUS TETES FIXES	51,55	25,36
h +INDEX NIVEAU 1 EN MEMOIRE	44,22	22,29

Tableau 8e: temps de traitement d'un fichier en indexé séquentiel.

La clé correspond généralement à une bijection entre les valeurs qu'elle prend et les articles: deux articles différents ont deux clés différentes. Certaines organisations permettent cependant les clés **dupliquées** (duplicated key, cas de RMS sous VMS), c'est-à-dire plusieurs articles avec une même valeur de la clé. Les clés **numériques** sont parfois acceptées.

La clé n'est pas forcément une zone contiguë de l'article, mais

peut être construite à partir d'un regroupement logique de plusieurs zones. On parlera alors de clés **segmentées** (cas de RMS). Certaines méthodes imposent cependant la contiguïté, et même l'obligation de commencer en début d'article.

Certaines organisations permettent d'utiliser des **clés multiples** et l'on parlera alors de clé **principale** (primary key, prime index dans VSAM), généralement non dupliquée, et de clés **secondaires** (secondary key, alternate key dans VSAM et RMS). Selon les méthodes d'accès il sera ou non possible de modifier la valeur des clés secondaires, d'avoir des clés secondaires "nulles" (aucune valeur n'est fournie pour une telle clé dans certains articles). Il peut y avoir interdiction de chevauchement entre clé primaire et secondaire. Pour optimiser l'espace occupé, les clés peuvent être compressées (front, en tête, rear, en queue dans VSAM, RMS, AIX dans VSAM). Les clés secondaires de VSAM peuvent être utilisées avec l'organisation séquentielle où une relation est établie entre valeur de clé et RBA.

Il convient bien entendu de ne pas confondre la structure hiérarchique formée pour accéder plus rapidement aux valeurs d'une clé et détaillées plus haut dans le cas de la clé primaire, avec la hiérarchie que l'on peut imaginer entre clé primaire et clés secondaires.

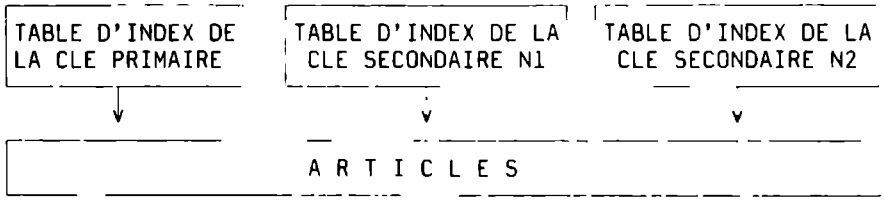


Figure 8f: chacune des clés dispose de sa propre structure à un ou plusieurs niveaux.

La lecture d'un article se fait généralement en donnant la clé **exacte**. Certaines méthodes d'accès permettent d'utiliser des clés **génériques** dans lesquelles seuls les premiers caractères de la clé recherchée sont donnés. Les clés **approximatives** permettent de fournir une valeur supérieure, supérieure ou égale, etc., de la clé recherchée.

Dans la méthode d'accès UFAS de GCOS3, les index sont stockés dans un fichier séparé: si les données sont stockées dans le fichier défini par $ FILE 3F,xxx par exemple, les index se trouveront dans le fichier $ FILE F3. Le séquentiel indexé (SI) de Prologue utilise aussi deux fichiers avec les suffixes -D et -I. Il en va de même dans le cas de VSAM avec la notion de "cluster" qui, pour l'organisation indexée, comprend deux composantes: la composante clé et la composante donnée. La notion d'«area» de RMS (cf. chapitre 6) est analogue.

[TI80A] donne des exemples comparatifs de manipulation de fichier indexé séquentiel (KIF) par des programmes écrits en Cobol sur un mini-ordinateur de Texas Instruments tournant sous le système d'exploitation DX10. On y voit l'effet, sur le temps de création, du

nombre de clés, l'incidence du tri préalable du fichier séquentiel qui sert à constituer le fichier indexé. On y constate aussi que les accès sont plus longs, d'où l'intérêt de "réorganiser" de temps en temps le fichier.

NOMBRE DE CLES	1	2	5	14
TEMPS DE CREATION				
.ENTREE TRIEE	1	1,3	2,6	46,7
.ENTREE NON TRIEE	2,1	4,1	12,5	53,0
ENTREE TRIEE				
.LECTURE SEQUENTIELLE	1	1	1	1
.LECTURE ALEATOIRE	1,4	1,4	1,4	1,4
.ECRITURE SEQUENTIELLE	2,6	2,6	2,9	3,7
.ECRITURE ALEATOIRE	3,3	3,3	3,7	4,7
ENTREE NON TRIEE				
.LECTURE SEQUENTIELLE	2,2	2,2	2,2	2,2
.LECTURE ALEATOIRE	2,3	2,3	2,3	2,3
.ECRITURE SEQUENTIELLE	4,1	4,1	4,4	5,4
.ECRITURE ALEATOIRE	4,4	4,4	4,8	5,8

Tableau 8g: exemples tirés de [TI80A].

L'organisation relative fait partie des organisations **directes** dans lesquelles les articles sont rangés à une adresse obtenue par calcul (on parlera de "hash-coding"). Dans les organisations directes, l'utilisateur dispose dans son programme d'un algorithme qui, en fonction du contenu de l'article, produit l'adresse à laquelle est stocké l'article. GCOS3 utilise un tel algorithme pour localiser une entrée (une identification d'utilisateur ou "userid") dans le répertoire du système (SMC): le nom de l'utilisateur est rangé dans deux mots (sur DPS8 un mot fait 36 bits), ces deux mots sont additionnés, le résultat est multiplié par 010101010101 et les bits 37 à 41 du résultat donnent le numéro du bloc (llink) où se trouve le "userid" recherché (un "llink" contient 25 userid).

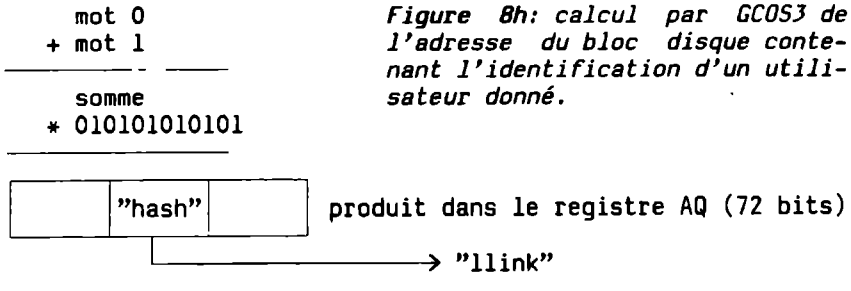


Figure 8h: calcul par GCOS3 de l'adresse du bloc disque contenant l'identification d'un utilisateur donné.

Comme l'indexé séquentiel, l'organisation directe est source de **collisions**: il y a collision lorsque plus d'un article doit être

stocké à la même adresse. Pour surmonter cette situation il faut gérer des zones de **débordement** (overflow area) dans lesquelles on rangera les articles qui n'ont pu être stockés à l'adresse "normale". Il en est de même en indexé: ainsi ISAM sous MVS réserve une piste par cylindre alloué pour ces débordements. Un bon algorithme se juge au petit nombre de collisions qu'il provoque. Les méthodes comme VSAM ou UFAS utilisent des zones libres au moment de la création du fichier pour faciliter l'évolution du contenu du fichier: de l'espace libre dans les CI, des CI libres. Dans RMS on parlera du "fill factor" (facteur de remplissage) pour exprimer le pourcentage de place libre réservée par "bucket" (équivalent du CI dans RMS).

[FRI81] donne un exemple comparant la lecture aléatoire d'un fichier en indexé VSAM (KSDS) optimisé avec celle d'un fichier en BDAM: la lecture en BDAM est plus rapide de 13 % sur un 3350, de 11 % sur un 3380.

Dans le monde IBM on entendra encore parler d'organisations **partitionnées** (partitioned). Un fichier partitionné est en fait un ensemble de fichiers séquentiels partageant le même espace disque. Ces "sous-fichiers" sont appelés "membres" (member) et généralement utilisés pour stocker des bibliothèques de programmes, de macros, de procédures. Seul le fichier est connu du SGF, les membres l'étant par la méthode d'accès. Dans les OS d'IBM le membre MEMB1 du fichier FICH s'écrira FICH(MEMB1). L'utilisation de ces fichiers partitionnés peut se révéler difficile dans la mesure où la gestion de l'espace libre est très statique: sous MVS par exemple, toute modification d'un membre conduit à sa réécriture complète et contiguë dans l'espace encore libre; l'espace libre une fois épuisé, il faut réorganiser le fichier avec les risques associés. Sous DOS/VSE il a fallu attendre la version SP2.1.0 (livrable en avril 85) pour disposer de l'extension dynamique lorsque les librairies sont pleines et pouvoir réutiliser la place occupée précédemment par des membres supprimés ou modifiés, encore que CICS utilise toujours l'ancienne version d'un programme quand une nouvelle version est cataloguée: il faut dans ce cas éviter de réutiliser l'espace occupé par l'ancienne version. Les fichiers gérés par LIB sous MS-DOS, LIB86 sous iRMX, relèvent de ce type d'organisation partitionnée.

8.2.2 – Organisations liées aux bases de données.

Nous emploierons le terme de base de données lorsqu'il y a des **relations** entre les articles (segments, record), relations **gérées** par la méthode d'accès, sous formes de pointeurs par exemple, pointeurs qu'il ne faut pas confondre avec ceux que nous avons vus dans le chapitre 5 au niveau de la gestion de l'espace disque. Ce sont deux niveaux totalement indépendants.

Le terme de base de données (data base) est plus récent que celui de fichier (file, data set), il correspond à une plus grande complexité dans l'organisation des données. Il est donc plus "porteur" et paraît plus alléchant pour les utilisateurs, d'où certains abus de la part des fournisseurs. Ainsi un gestionnaire de données très répandu,

et très apprécié à juste titre, sur les PC offre une organisation indexée séquentielle avec des index qu'il faut régénérer explicitement lorsque l'on a modifié des articles, ce qui ne se produit pas dans le cas des indexés séquentiels des moyens ou gros ordinateurs. Ce logiciel porte cependant le nom de "base de données", DBase II (Data Base II), à ne pas confondre avec DB2 d'IBM.

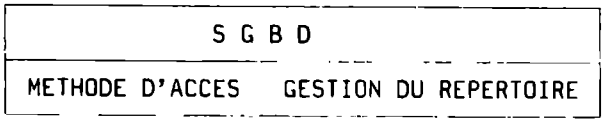


Figure 8i: SGBD bâtis au-dessus des fichiers classiques.

D'autres méthodes d'accès aussi appelées SGBD correspondent à une couche supplémentaire ajoutée au-dessus des méthodes d'accès fichier et du système de gestion de fichiers (figure 8i). Ces SGBD créent des liens entre des fichiers classiques et permettent ainsi d'offrir certains des avantages des bases de données. C'est par exemple le cas du SGBD TOTAL de Cincom, une implémentation analogue est appelée IMAGE chez HP. Dans Prologue, la base de données (BD) est ainsi construite au-dessus du multi-critères (MC) lui-même bâti au-dessus du séquentiel indexé (SI):

NGF → SI → MC → BD → utilisateur

Quels sont les avantages des bases de données par rapport aux fichiers classiques? L'avantage principal est de mieux séparer les fonctions de **traitement** des fonctions de **stockage**, et plus généralement de gestion pure des données. Tout ce qui décrit les informations se trouve dans la base de données et non pas dans les programmes qui les traitent. Cette **indépendance** accrue se traduit par les éléments suivants:

- indépendance de la **structure** des articles: un programme X verra l'article ART1 (réellement formé des cinq zones ART1A, ART1B jusqu'à ART1E) comme étant composé des deux zones (rubrique, champ, field) ART1A et ART1C, alors que le programme Y le verra comme comprenant les zones ART1A, ART1B et ART1D. Ceci correspond aux notions de **schéma** et **sous-schéma** des bases de données CODASYL;
- indépendance des **formats**: un nombre sera ainsi stocké sous une forme propre au SGBD et non pas liée à tel ou tel programme d'application. Le SGBD stockera par exemple les nombres sous une forme décimale, une chaîne de caractères donc; l'application A écrite en Cobol le verra en décimal étendu, l'application B écrite en PL/I le verra en décimal condensé (voir tome I), l'application D en Fortran le manipulera en flottant, l'application E écrite en langage C en fera un entier binaire. Toutes les conversions nécessaires sont réalisées par le SGBD;
- meilleure **intégrité**, **cohérence** des données puisque le SGBD dispose de toutes les caractéristiques des données et de tous les moyens pour vérifier leur validité, corriger les erreurs, les incidents;
- meilleure **sécurité**: le SGBD a les moyens de contrôler l'accès à la

base de données.
En dehors de cette indépendance entre stockage et traitement, les SGBD offrent d'autres avantages:
- des **organisations** plus complexes qui permettent de modéliser des réalités plus compliquées,
- un moindre **redondance** des données dans la mesure où il n'est plus utile de reproduire la même occurrence d'une information dans plusieurs fichiers, seule possibilité qu'avaient les applications de créer des relations entre les articles;
- des **outils intégrés** de saisie (data entry), d'interrogation (query), d'édition d'états (report).

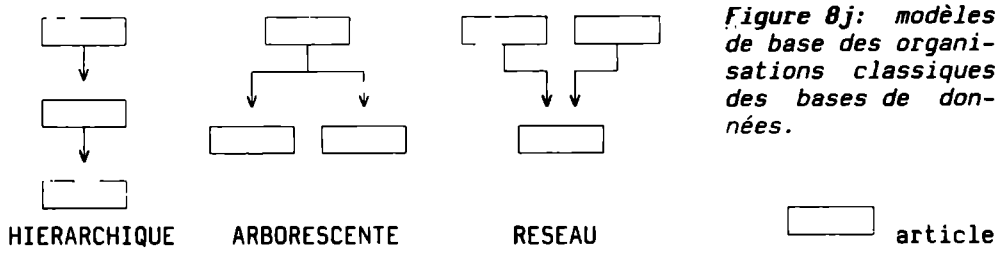


Figure 8j: modèles de base des organisations classiques des bases de données.

Les organisations les plus fréquentes sont les structures hiérarchiques, réseau et relationnelles.

Dans la structure **hiérarchique** un article "propriétaire" pointe vers plusieurs articles "membre", tout comme un répertoire pointe vers les fichiers qui y sont contenus. Un membre peut à son tour être le propriétaire d'autres membres, d'où le nom de hiérarchie ou d'arborescence. Un membre n'a qu'un seul propriétaire. DL/I d'IBM appartient à cette catégorie de bases de données.

La structure en **réseau** (network) peut être considérée comme un surensemble de la précédente dans la mesure où un membre peut avoir plusieurs propriétaires. IDS (Integrated Data Store) de Bull est l'archétype de cette catégorie qui a fait l'objet d'une standardisation CODASYL et se trouve sur la plupart des systèmes informatiques: IDS II chez Bull, IDMS de Culliname sur les systèmes IBM, MIDS sur Multics. etc.

La structure **relationnelle** (relational) est plus récente et tend à supplanter les autres organisations, bien qu'elle soit plus gourmande en puissance de traitement entre autres. Les données sont organisées sous forme de fichiers plats ou "tables", chaque table correspondant à une "relation". De nouvelles relations peuvent être construites à partir des relations existantes. Dans cette catégorie figurent DB2 chez IBM, MRDS sur Multics.

L'organisation des bases de données est définie en utilisant un langage de description génériquement appelé DDL (Data Description Language), le DML (Data Manipulation Language) correspondant à l'accès aux données.

Nous évoquerons dans le chapitre sur les fonctions réseau les bases de données **distribuées**.

8.3 – Les modes d'accès.

Une fois le fichier ou la base de données défini, il est possible d'y accéder, pour ranger, modifier, chercher des informations.

Au niveau des fichiers classiques, le langage de gestion le plus répandu, Cobol, reconnaît aujourd'hui trois modes d'accès standardisés par le CODASYL: séquentiel, aléatoire, dynamique.

MODE D'ACCES	ORGANISATION SEQ	REL	INDX
SEQUENTIEL	X		
ALEATOIRE	X	X	X
DYNAMIQUE	X	X	X

Tableau 8k: modes d'accès permis (X) selon les organisations. Ce type de tableau est couramment fourni par les constructeurs.

Dans l'accès **séquentiel**, les articles sont obligatoirement lus dans l'ordre où ils ont été écrits, c'est-à-dire dans l'ordre où ils sont rangés dans le fichier.

Avec l'accès **aléatoire** (random) tout article peut être accédé indépendamment des autres, directement en donnant son numéro (organisation relative), sa clé (organisation indexée séquentielle) ou son pointeur dans la base (IDS II). L'organisation séquentielle interdit bien entendu ce mode d'accès. L'aspect "direct" de l'accès correspond à la vue qu'en a l'utilisateur, dans le cas de l'indexé séquentiel nous avons vu que l'accès à l'article requérait une recherche dans une ou plusieurs tables d'index. RMS offre aussi la possibilité d'accéder à un article par son adresse RFA (Record's File Address) dans le fichier.

L'accès **dynamique** correspond à un positionnement "direct" sur un article (verbe START du Cobol 74), suivi d'accès séquentiels (verbe READ NEXT du Cobol 74) pour obtenir les articles suivants. Là encore l'organisation séquentielle ne permet pas cet accès.

ORGANISATION	IBM	GCOS3	PRIMOS	VMS
SEQUENTIELLE	SAM BSAM QSAM SAM-E VSAM/ESDS	GFRC UFAS	SAM	RMS
INDEXEE SEQ.	ISAM BISAM QISAM VSAM/KSDS	ISP UFAS	MIDAS	RMS
RELATIVE	VSAM/RSDS	UFAS		RMS
DIRECTE	DAM BDAM		DAM	
PARTITIONNEE	PAM BPAM			

Tableau 8l: exemples de méthodes d'accès dans divers SE.

Le choix du mode d'accès est fait par la clause ACCESS MODE IS en Cobol 74. Certaines organisations permettent plusieurs modes d'accès:

traiter tous les articles d'un fichier organisé en indexé séquentiel avec un mode d'accès séquentiel sera plus rapide et moins coûteux en temps processeur qu'avec un accès aléatoire; ce dernier sera plus adapté au traitement d'une partie d'un fichier.

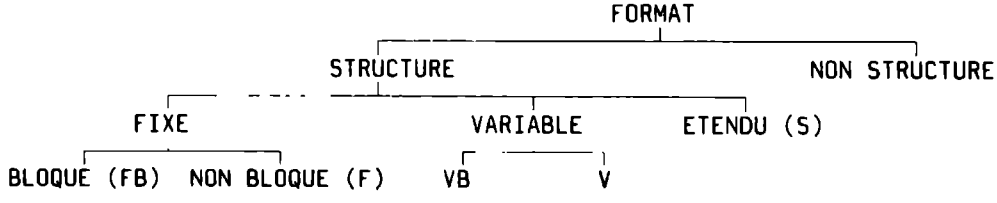
Les SE présentent fréquemment aujourd'hui deux ensembles de méthodes d'accès:

- les "anciennes" qui précédent la normalisation Codasyl: SAM, ISAM chez IBM, GFRC, ISP avec GCOS3;
- les "nouvelles" qui prennent en compte les normes Codasyl: VSAM chez IBM, UFAS chez Bull.

8.4 – Relations entre bloc et articles.

Le bloc étant l'unité de transfert entre mémoires périphérique et centrale, l'article l'unité de traitement, il se produira différentes situations entre les tailles de ces deux entités. Ces relations entre article et bloc sont prises en compte par les méthodes d'accès. Lorsque la taille de l'article est inférieure à celle du bloc, ce dernier contient un ou plusieurs articles et l'on parlera de **blocage** des articles: IBM utilisera les sigles FB (fixed blocked) pour des articles et des blocs de longueurs constantes, VB (variable blocked) si les blocs contiennent plusieurs articles de longueurs variables, F et V s'il n'y a pas blocage. Ce sigle est la valeur du paramètre RECFM (RECord ForMat) de la commande // DD. La norme américaine X3.27 définit un format pour les articles variables sur bande avec 4 octets rajoutés en tête (format appelé D sur VMS, le format V qui ajoute deux octets étant utilisé sur disque). RMS offre aussi un format appelé VFC (Variable avec zone de Contrôle Fixe) qui permet, par exemple, de mettre des numéros de ligne dans la zone de contrôle.

Dans le cas où l'article est plus long que la taille du bloc IBM utilisera le terme de "spanned" (à cheval, étendu) abrégé en S.



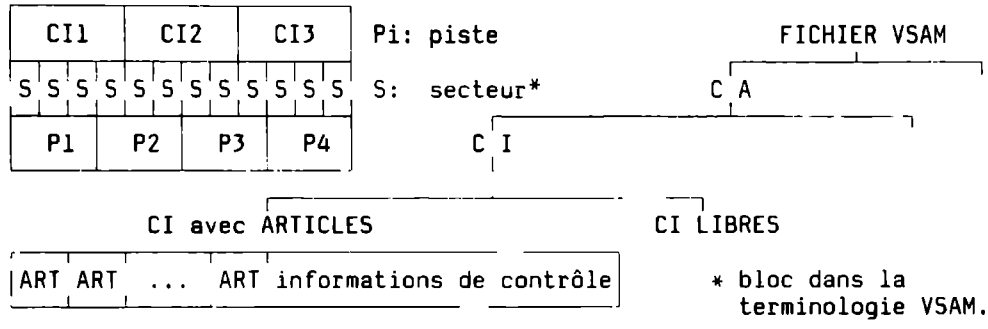
Le format "flot" (stream) ou **non structuré** parce que les articles ne sont pas découpés en zone bien définies, correspond donc à une absence de format. Les "articles" y sont délimités par un caractère de fin (terminator), couramment RC (Retour Chariot) ou NL (Nouvelle Ligne), ou les deux. C'est ce type d'organisation que met en oeuvre l'instruction LINE INPUT en Basic, c'est la seule organisation offerte par Unix ou MS-DOS au sens strict; les fichiers générés par les éditeurs de texte de MS-DOS, le Basic, ont leurs articles (lignes) bornées par RC-NL (CR-LF en américain, 0D0A en hexadécimal). La bureautique, domaine du traitement de document, du traitement de

texte, est par opposition à l'informatique classique, un domaine non structuré du point de vue de l'organisation des fichiers.

Figure 8m: bloc de deux articles de longueur variable.

LBLOC contient la longueur du bloc
LARTi contient une longueur d'article.

Pour un fichier VSAM nous avons bien entendu la notion d'article, le CI (Intervalle de Contrôle) correspond à la notion de bloc, d'unité de transfert entre mémoire périphérique MP et mémoire centrale MC. Le CI a une longueur fixe dans un fichier. Si le CI correspond à la notion de bloc, indépendamment du type de disque utilisé (CKD, FBA dans le tome III), le CA (Control Area) constitue un cylindre logique, indépendant du type de disque.



8.5 – Compression des données.

Afin d'augmenter la quantité d'information rangée sur les supports de stockage, les disques principalement, des mécanismes diminuant le nombre de bits réellement stockés peuvent être mis en oeuvre. Le mécanisme le plus simple est la compression des caractères consécutifs: AAAAA sera ainsi stocké sous la forme d'une séquence de type R5A, le code R indiquant une répétition de cinq caractères A. Ceci est souvent utilisé pour les espaces et c'est ce que réalise par défaut Primos. On parle alors de compression **horizontale**. Souvent plus efficace pour les fichiers structurés, la compression **verticale** permet de ne stocker que ce qui est modifié entre deux articles consécutifs d'un même fichier. La compression des clés en indexé séquentiel relève de méthodes analogues.

chapitre 9

communication et synchronisation entre processus

Les processus et le système d'exploitation sont amenés à communiquer entre eux, **directement** pour signaler l'arrivée de certains événements, comme les fins d'E/S, pour échanger des informations, **indirectement** lorsqu'ils veulent accéder à des ressources qui ne supportent qu'un nombre réduit d'accès simultanés, accès pour lesquels les processus doivent s'accorder **explicitement** ou être coordonnés **implicitement** par le SE. Le chapitre 14 traitera plus particulièrement ce problème d'accès concurrent, nous aborderons par contre la communication dans ce chapitre.

Lorsque l'on parle de communication entre deux entités on distingue les communications **synchrones**, analogues au téléphone, et les communications **asynchrones**, ou "sans connexion", analogues à la poste. La différence réside dans l'établissement préalable d'un chemin dans le cas synchrone: il y a ouverture d'un chemin, attente de l'acceptation du partenaire, puis dialogue ensuite. Le dialogue une fois terminé, le chemin est rompu.

9.1 – Communication avec le SE.

La principale raison de communiquer avec le SE est représentée par les requêtes que formulent les processus, requêtes sollicitant les **services** du SE. Ces services sont accessibles par le langage de commande (JCL, OCL), et par une interface programmatique, celle qui nous intéresse plus particulièrement ici. [VMS85B] correspond à cette interface et porte d'ailleurs le nom de "System Services Reference Manual". De même la SVID pour Unix, [RTE83C] dans le cas de RTE-A, [RTE83D] correspondant au langage de commande externe.

La forme de ces requêtes diffère d'un système à l'autre, avec quelques grands cas généraux:

- l'**appel**: l'utilisateur appelle le système d'exploitation avec une instruction machine d'appel (CALL). On trouve cette méthode dans CP/M, dans GCOS7, avec une réalité matérielle beaucoup plus complexe dans le second cas;
- l'**interruption** représente une autre forme d'appel, avec l'avantage de rendre l'appelant indépendant des adresses mémoire où sont stoc-

kés les programmes du SE: dans une instruction d'appel il faut préciser l'**adresse** de l'appelé, donc savoir où il se trouve et lui être **lié**; dans l'interruption c'est un **numéro** de fonction qui est donné, l'association entre numéro plus paramètres fournis par le demandeur et la fonction sollicitée est faite par des tables et/ou du code intermédiaire. L'interruption a de plus l'avantage, sur des machines moins complexes que des DPS7, de mettre en oeuvre le mécanisme de protection déclenché par l'interruption: l'appelé est ainsi mis en mode maître (privilégié) automatiquement. Les DOS et OS d'IBM utilisent cette méthode d'interruption "programme" (instruction SVC), de même MS-DOS (instruction INT), GCOS3 (instruction MME), etc.

A titre d'anecdote il est intéressant de signaler des fréquences maximum d'appel au SE préconisées par les constructeurs: [SCH80] recommande ainsi de ne pas dépasser 500 SVC par seconde sur un 158 ou un 3031, 1200 sur un 168 ou un 3032, 2000 sur un 3033.

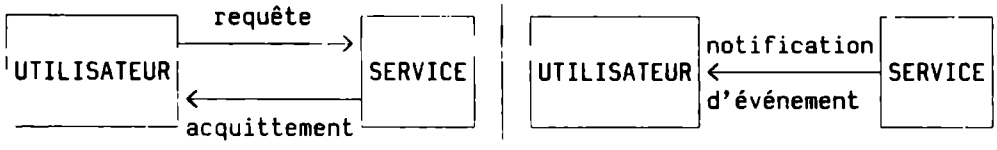


Figure 9a: types de communication entre utilisateur et SE.

9.2 – Communication inter-processus.

La communication directe entre processus comprend:

- l'échange de **signaux** de synchronisation, d'acquittement, de notification d'événements, au moyen de macros du type POST, SIGNAL (création d'événement), KILL (création d'un signal sous Unix) et WAIT (attente d'événement). Ces macros utilisent des:
  * **sémaphores** inventés par Dijkstra (cf. tome I) avec les opérations V (création d'événement, incrémentation du compteur du sémaphore) et P (attente d'événement et décrémentation du même compteur quand l'événement arrive). Sur DPS7 les sémaphores existent au niveau du matériel dans l'architecture externe, sur iRMX86 ils sont créés de toute pièce par le SE et peuvent être incrémentés/décrémentés de n unités à chaque référence. Ils sont utilisés entre autres pour les problèmes d'exclusion mutuelle;
  * **drapeaux** (flags, events): un bit représente l'apparition (valeur 1) de l'événement auquel il est associé ou l'absence de cet événement (valeur 0). Le VAX n'utilise pas les sémaphores mais ces drapeaux (cf. exemple les utilisant dans le chapitre 6). Les drapeaux se prêtent plus facilement aux attentes d'événements multiples (sémaphores en séquence) ou à celles d'un événement parmi plusieurs événement possibles;
- l'échange d'**informations** au moyen de:
  * **fichiers**: ils sont plutôt utilisés pour la communication entre étapes, travaux, moins pour la communication interprocessus, pour des raisons de performance principalement;

* fichiers en mémoire: **tubes** (pipe) d'Unix (entre commandes consécutives, cmd1|cmd2, aussi bien qu'entre processus), d'OS9, mécanisme synchrone et unidirectionnel dans lequel un processus écrit et un autre lit. Dans iRMX86 on parlera de fichiers "flot" (stream) pour les tubes, les fichiers sur disque étant dits "nommés" (named);
* **boîtes aux lettres** (BàL, mailbox) de VMS, iRMX qui offre la possibilité d'envoyer un acquittement à l'émetteur. Dans RTE-A qui parle de "class I/O" les BàL sont implémentées dans la zone de communication du système appelée SAM. Ce mécanisme est asynchrone. iRMX offre le choix entre la gestion FIFO ou la gestion par priorité;
* **sémaphores** avec messages: GCOS7, iRMX avec gestion FIFO ou par priorité;
* **mémoire partagée** (commune) offerte par la structure du matériel ou par la structure de l'adressage, en mémoire virtuelle ou en mémoire centrale. RTE-A dispose ainsi de zones partageables appelées "system common area" et EMA. GCOS3 offre un service appelé INTERCOM: il gère une zone mémoire dans laquelle il recopie le message de l'émetteur pour le recopier ensuite dans la mémoire du destinataire. Le SE crée ainsi une mémoire partagée directement (RTE-A), ou indirectement (GCOS3). On trouve une solution de ce type sur de nombreux SE, sinon tous, ce mécanisme est dit "standard" dans MVS;
* fonctions **réseau**: les processus connus du réseau peuvent ainsi dialoguer à l'intérieur d'un même ordinateur. C'est le cas de deux applications de VTAM (c'est-à-dire faisant appel à VTAM en respectant l'API, Application Program Interface) qui s'échangent des messages à travers VTAM, à savoir le gestionnaire réseau dans le monde IBM.

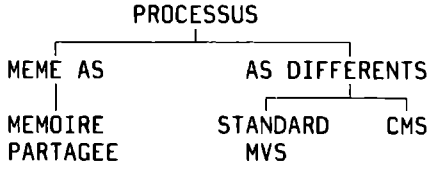


Figure 9b: les types de communication entre processus dans MVS selon [MAR86].

CMS: Cross Memory Services

Le mécanisme standard de MVS repose ainsi sur l'utilisation de la région commune de chaque espace d'adresses, mais cette région est vite saturée, le matériel du 370 ne dispose que de 16 clés de protection (8 seulement en mode non privilégié), et le coût processeur est élevé:

1. l'émetteur acquiert un tampon en zone commune,
2. il y copie le message,
3. il envoie la requête via une tâche SRB,
4. il attend que le destinataire acquitte ("post") le message.

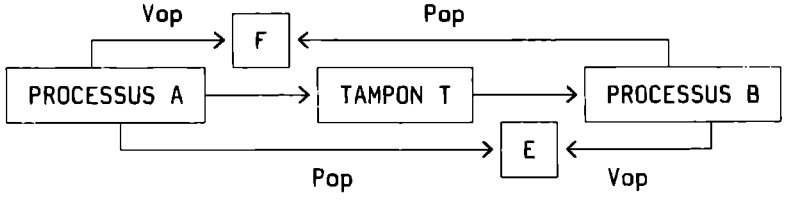
Le distributeur est comme on le voit, mis à contribution, d'où le coût élevé. Pour améliorer ceci et diminuer la contrainte des 16 Mo adressables en mémoire virtuelle, IBM a procédé à une évolution architecturale sur les 308X, appelée DAS (Dual Space Address) au niveau du

matériel, et offerte sous le nom de CMS (Cross Memory Services) au niveau de MVS; JES3 et IMS peuvent ainsi diminuer la taille de leur zone CSA ([SMI82]); des sous-systèmes comme DB2 logent leurs volumineuses routines dans un second AS.

Les SE comme MVS, orientés vers le traitement par lot ont généralement été conçus sans avoir pour objectif la communication entre processus, et les solutions qu'ils ont mis en oeuvre sont souvent lourdes.

9.3 – Exemple.

Considérons deux processus A et B se partageant un même tampon T. Le processus A alimente (remplit) le tampon, le processus B le lit lorsqu'il est plein. Pour se synchroniser, c'est-à-dire pour savoir quand le tampon T est plein, un sémaphore F est utilisé. Le sémaphore E signifie au processus A que B a libéré le tampon:



Les deux processus déroulent les algorithmes ci-dessous:

PROCESSUS A	PROCESSUS B	TAMPON T	SEMAPHORE F	SEMAPHORE E
		vide	0	0
1. remplit T		plein	0	0
	2. Pop sur F mis en attente	plein	-1	0
3. Vop sur F		plein	0	0
	4. lit le tampon	vide	0	0
5. Pop sur E mis en attente		vide	0	-1
	6. Vop sur E	vide	0	0
7. reprend en 1.		vide	0	0
	8. reprend en 2.	vide	0	0

Vop: opération V (création d'événement, plus 1 sur le compteur).
Pop: opération P (attente d'événement, moins 1 sur le compteur).

chapitre 10

évolution des systèmes d'exploitation

Nous avons maintenant une bonne vue d'ensemble des fonctions et de la structure générale des systèmes d'exploitation. Une étude de leur évolution donnera un éclairage complémentaire et servira d'introduction aux prochains chapitres.

10.1 – SE et travail lus ensemble.

Initialement le système d'exploitation, ou ce qui en tenait lieu, était lu avec le travail qu'il devait exécuter. Cette technique ancienne a comme **avantages**:

- une gestion simple de la mémoire centrale, facile à comprendre,
- aucun support matériel particulier n'est nécessaire. Un mécanisme de protection est cependant souhaitable afin que le travail ne détruise pas le SE.

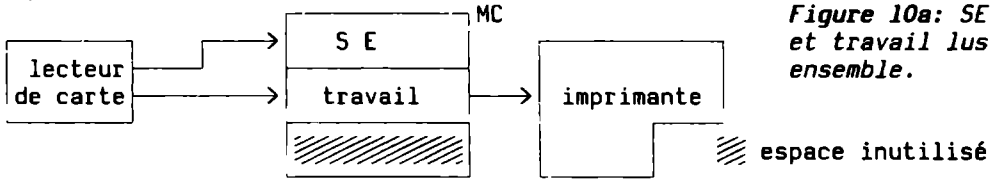


Figure 10a: SE et travail lus ensemble.

Au titre des **défauts** nous aurons:

- une mauvaise utilisation de la mémoire, car tous les travaux ne la saturent pas, seuls les plus gros l'occupent totalement,
- une mauvaise utilisation du processeur:
  * pendant la lecture du SE,
  * pendant la lecture du travail,
  * pendant les E/S du travail,
  * pendant l'impression des résultats,

  d'autant plus néfaste que les appareils de lecture/écriture utilisés sont lents,
- la taille maximum d'un travail est limitée par la taille mémoire qui reste disponible, une fois le SE chargé,
- la mémoire contient des données qui ne sont jamais référencées.

10.2 – SE résident.

Avec le SE résident l'amélioration porte sur la suppression des temps morts provoqués par la lecture du SE. Les SE offrent de plus en plus de fonctionnalités, ils grossissent de façon continue et il est de plus en plus difficile de les conserver totalement en MC. La figure 10b illustre la structure d'un tel SE: on voit apparaître le **moniteur d'enchaînement** dont la fonction est d'enchaîner l'exécution des travaux, l'un après l'autre.

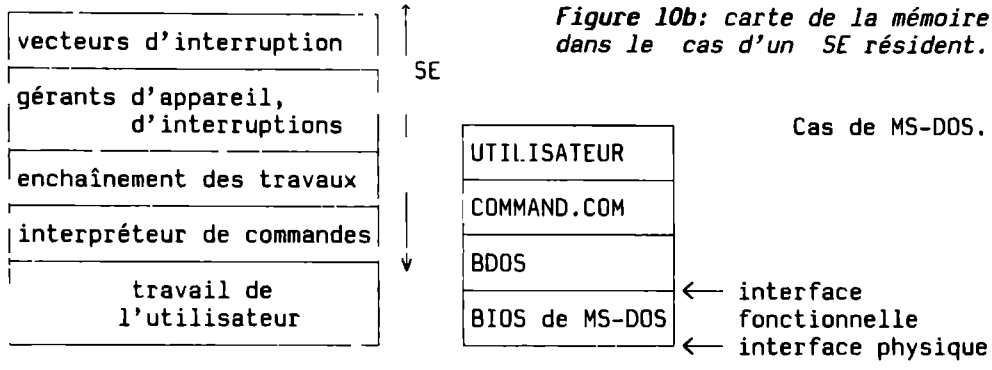


Figure 10b: carte de la mémoire dans le cas d'un SE résident.

Des exemples de tels SE sont actuellement CP/M et MS-DOS où les gérants d'appareils (drivers, handlers) se trouvent dans le BIOS qui assure au BDOS une interface indépendante de la nature physique des appareils. Dans le cas de CP/M la figure 10c présente la structure générale de la carte mémoire (memory-layout).

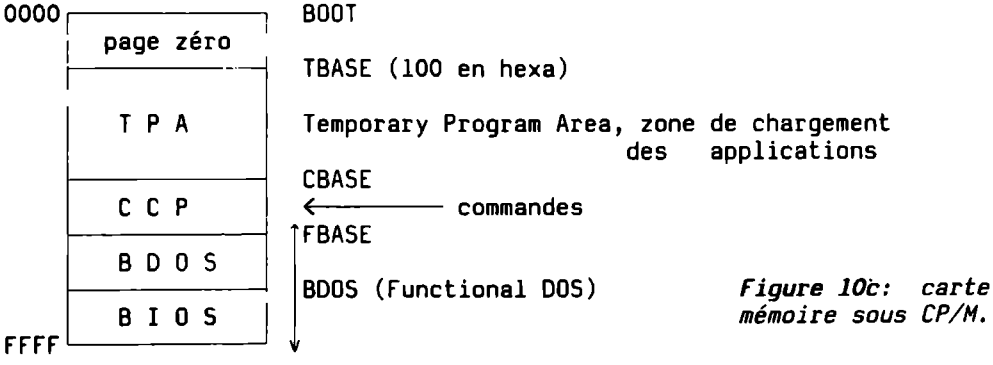


Figure 10c: carte mémoire sous CP/M.

Un programme utilisateur (de même que les commandes dites transitoires, transient) est chargé de TBASE à CBASE ou FBASE (dans ce dernier cas le CCP est écrasé). Le CCP (2 Ko environ) est l'analyseur de commandes; il contient aussi les commandes résidentes (DIR, ERA, REN, SAVE, TYPE); les commandes externes aussi appelées "non résidentes" correspondent à des fichiers de type COM (par exemple PIP.COM pour l'utilitaire de copie). Le BIOS occupe 0,5 à 1 Ko et con-

tient les gérants d'appareil qui sont susceptibles de changer d'un modèle d'ordinateur à l'autre (CP/M tourne sur plusieurs types de micro-ordinateurs). Le BDOS occupe environ 3 Ko. Cette caractéristique de résidence ou non-résidence des commandes illustre une nouvelle fois la difficulté de tracer une frontière claire entre logiciel de base, système d'exploitation et applications: la commande COPY de MS-DOS est résidente, DISKCOPY ne l'est pas, mais pour beaucoup toutes deux font partie intégrante de MS-DOS.

La figure 10d montre la carte mémoire de MS-DOS. FLEX-9 conduit à une carte mémoire analogue à celles de CP/M ou MS-DOS. PCP (Primary Control Program), l'un des premiers OS utilisés sur les 360 d'IBM relève aussi de cette catégorie.

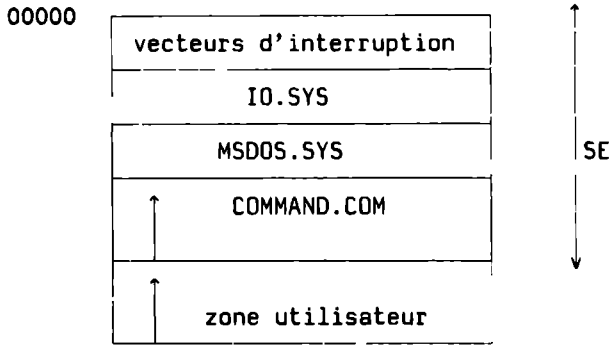


Figure 10d: carte mémoire dans le cas de MS-DOS.

On note dans les deux exemples ci-dessus que CP/M charge les programmes dans les adresses "basses", le SE résidant dans les adresses "hautes", alors que MS-DOS travaille d'une façon opposée. La raison de cette différence réside certainement dans le fait que les appels aux fonctions de CP/M sont faites par appels de sous-programme alors que celles de MS-DOS sont appelées par interruptions, et surtout parce que les programmes ne sont pas "relogeables" (translatables en MC, cf. tome V) sur un Z80, alors qu'ils le sont avec un 8086: il suffit d'y changer la valeur des registres de base des segments CS, DS, etc. Dans CP/M les programmes sont chargés à l'adresse TBASE, et cette adresse n'est pas remise en cause si le SE voit sa taille augmenter: il croîtra dans les adresses hautes.

10.3 – SE avec zones de recouvrement.

Cette nouvelle évolution du SE a pour but de résoudre le problème posé par l'augmentation de la taille du SE. Ce dernier est décomposé en plusieurs parties, l'une appelée le **résident** ou noyau, est constamment en MC, les autres parties sont stockées sur mémoire secondaire, et amenées en MC quand elles sont nécessaires, dans des zones de recouvrement du système (system overlay, transient area, SSA pour System Slave Area sous GCOS3). Soulignons au passage le fait que le terme de **noyau** est associé au caractère de résidence en mémoire, ce qui n'est pas exactement le cas pour un terme qui lui est proche, le terme

de superviseur. Les premiers DOS de la série 360 d'IBM avaient un noyau d'environ 6 Ko (1300 instructions), la zone des recouvrements occupait 1700 octets (400 instructions environ), ce qui laisse rêveur quand on compare avec les MS-DOS actuels. SIRIS7 de l'ex-CII avait un noyau de 70 Ko et une zone de recouvrement de 10 Ko.

Le disque devient alors indispensable puisque c'est à partir de lui que les modules non résidents seront chargés en mémoire centrale. Le choix des modules résidents se fait selon des critères:
- de bon sens: les modules de gestion des E/S disques, des interruptions doivent être résidents par exemple,
- de taille des modules,
- de performance.

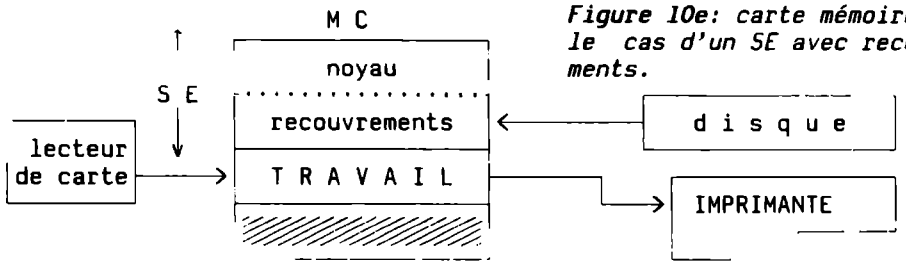


Figure 10e: carte mémoire dans le cas d'un SE avec recouvrements.

La résidence de certains modules est imposée par le constructeur. Des SE permettent à l'utilisateur d'ajouter des modules dans la liste des modules résidents: d'une installation à l'autre, en fonction de la charge réelle, la liste de ces modules variera et ce sera l'une des tâches de la cellule système que de définir les modules qu'il convient de rendre résidents. MCOUNT sur GCOS3 permet ainsi de connaître le nombre d'appels à chaque module de GCOS, les plus souvent appelés étant des candidats à la résidence. Le système est fréquemment coupé en deux et rangé aux deux extrémités de la mémoire (figure 10f) afin d'avoir deux possibilités d'expansion indépendantes. C'était le cas d'un des premiers SE d'IBM: OS/360 PCP. On retrouve cette structure sous MVS au niveau de la mémoire virtuelle.

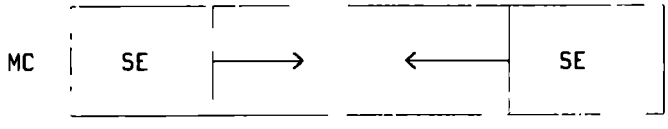


Figure 10f: SE chargé aux deux extrémités de la mémoire.

10.4 – Recouvrement des E/S lentes (SPOULE).

Cette fonction, réalisée par le GPL, a pour objectifs:
- d'éviter qu'un travail en MC ne fasse directement des E/S sur des périphériques aussi lents que des lecteurs de cartes ou des imprimantes,
- de rendre ces périphériques partageables par plusieurs utilisateurs,
- de recouvrir les E/S sur périphériques lents avec le déroulement du travail.

On voit apparaître la notion de **disque système**, disque sur lequel sont

stockés le SE, les files d'entrée, de sortie.

[FRE76] signale que l'introduction du GPL (SPOOL) sur l'IBM 1401 permit d'accroître le taux d'utilisation du processeur de 20% à 50%; dans les 50%, 10% étaient consommés par le GPL. Sur IBM 360 la fonction GPL ne fut pas implémentée dès le départ et il y eut plusieurs réalisations particulières comme HASP (Houston Automatic Spooling Priority, évolution des "system reader" et "system writer") sous OS/MFT ou OS/MVT. HASP était estimé plus performant, en particulier parce qu'il n'interprétait pas le JCL lors de la lecture des travaux, mais seulement lorsqu'il soumettait un travail pour exécution, c'est-à-dire lorsqu'il l'inscrivait dans la file des travaux du système (system job queue). Sous OS/MVS, HASP est devenu JES2, comme ASP est devenu JES3.

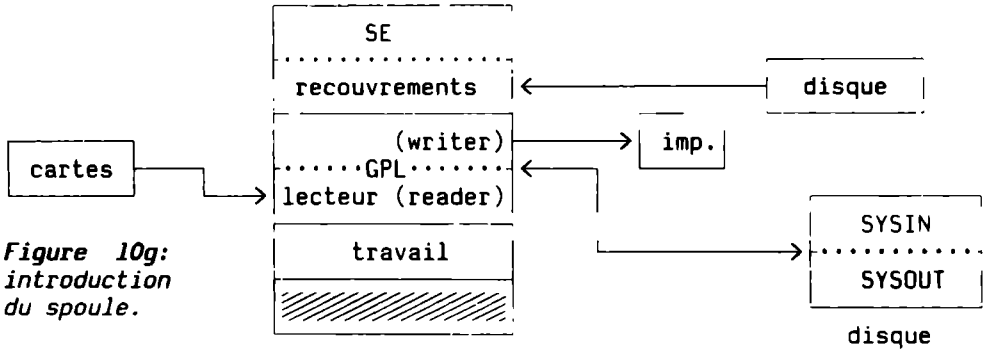


Figure 10g: introduction du spoule.

10.5 – Multiprogrammation.

La multiprogrammation correspond au déroulement simultané de plusieurs **travaux indépendants** (independent jobs) sous le contrôle et la protection du système d'exploitation. Elle a pour principal objectif de tirer le rendement maximum des ressources du système. La principale conséquence est une complexité accrue du SE.

Pour être viable la multiprogrammation exige un mécanisme de protection des travaux entre eux et vis-à-vis du système d'exploitation:
- registres base et limite sur DPS/8, 1100, CYBER, DEC10,
- clés sur IBM/370,
- mémoire virtuelle avec un espace d'adresses distinct par processus,

par exemple. On parlera de **taux de multiprogrammation** (multiprogramming depth, level of multiprogramming) pour se référer au nombre de travaux se déroulant en parallèle.

10.5.1 – SE multitâches (multitasking).

Un système avec multiprogrammation offre un partage dynamique du processeur au niveau des travaux: il y a simultanément plusieurs travaux en cours d'exécution dans la machine.

La fonction "multitâche" intervient au niveau inférieur et constitue un deuxième niveau de concurrence: chaque étape peut en fait être composée de plusieurs processus (ou tâches). On trouve dans

[WOL81] la définition suivante: "Aptitude d'un SE à gérer plusieurs processus d'un programme, ces processus s'exécutent simultanément. Le traitement multitâches est un sous-ensemble important de la multiprogrammation. Ces processus doivent se partager des données et ont donc besoin d'une communication interprocessus.". Ce "multitasking" peut être réalisé:

- au niveau d'une étape: c'est l'étape elle-même qui répartit son temps processeur entre ses différents processus que le superviseur ne "voit" donc pas. C'est le cas par exemple de GCOS3 dans les environnements transactionnels TDS et temps partagé TSS; c'est aussi le cas pour CICS chez IBM (rôle du module "gestion des tâches" dans la figure 10n);
- directement par le superviseur qui "voit" directement tous les processus. On parle alors de "multitasking" **généralisé**. l'attribution du temps processeur aux transactions. Les DOS et OS d'IBM offrent ce fonctionnement multitâche généralisé, de même que GCOS7.

Le "multitasking" a été introduit dans les OS d'IBM avec MVT, MFT n'en disposait pas; la macro ATTACH permet d'y créer une tâche.

Dans les SE temps réel la notion de travail n'existe pas vraiment, l'ensemble de la charge constitue le "travail" et on y parlera plutôt de "multitasking" que de multiprogrammation.

Comme souvent en informatique, la définition des termes est imprécise et il n'est pas rare qu'il y ait confusion entre multiprogrammation et multitâche.

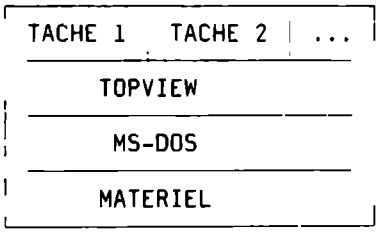


Figure 10h: fonctionnement multitâche créé par un logiciel multifenêtres.

Le "multitasking" peut aussi être créé par un élément externe au SE ou présenté comme une extension du SE. C'est le cas des logiciels **multifenêtres** qui comme Topview tournent "au-dessus" de MS-DOS pour permettre à l'utilisateur d'exécuter plusieurs tâches, avec une fenêtre associée à chacune d'elles sur l'écran du micro-ordinateur. MacBasic crée ce fonctionnement multitâches dans l'environnement Basic, le SE ne voyant qu'une seule tâche.

10.5.2 – Système multi-utilisateurs.

Les micro-ordinateurs et les postes de travail bureautiques ont popularisé le terme de mono/**multi-utilisateurs**. Un SE mono-utilisateur ne permet qu'à un seul utilisateur de travailler à un moment donné. Un exemple en est MS-DOS dans lequel la notion d'utilisateur n'est pas réellement implémentée: toute personne pianotant sur le clavier est considérée comme étant l'utilisateur. CP/M, CP/M-86 ont une notion

d'utilisateur, un seul étant actif à un moment donné; les utilisateurs sont numérotés de 1 à 15, et les fichiers sont associés à un utilisateur (le numéro zéro est réservé pour les fichiers accessibles à tous).

Mono-utilisateur, MS-DOS est monotâche encore en version 3, l'impression simultanée réalisée par la commande PRINT ne correspond pas à du "multitasking" puisque particulière à un périphérique bien particulier.

Un SE multi-utilisateur accepte simultanément plusieurs utilisateurs envoyant des commandes, indépendamment les uns des autres. On retrouve ici la définition de la multiprogrammation et nous admettrons par la suite que mono/multi-utilisateurs est équivalent à mono-multiprogrammation.

Le terme de mono/**multi-postes** correspond plutôt à une caractéristique physique d'une configuration qu'à une propriété du système d'exploitation. On donnera comme exemple le cas de machines multipostes où tous les utilisateurs n'ont accès qu'à un seul programme ou qu'à une seule fonction (cf. les multiclaviers).

Une enquête IDC menée sur le marché français donne pour les systèmes multipostes, en excluant les réseaux locaux, un marché de 2,4 milliards de dollars en 1983 et une estimation de 2,8 milliards pour 1984, soit une croissance de 16 % par an qui devrait se maintenir. Dans nombre de domaines d'application il n'est pas évident de faire le choix entre un mini et un réseau local de PC, encore qu'il faille se méfier de la lenteur de nombre d'entre eux, même en 1986.

SE	MONO-UTILISATEUR	MULTI-UTILISATEUR
MONOTACHE	MS-DOS CP/M CMS FLEX	GCOS3
MULTITACHE	CCP/M Pick II Amigados P/OS (DEC)	DOS MVS GCOS VMS Primos iRMX Prologue Unix MP/M-86

Figure 10i: exemples de SE classés selon le nombre d'utilisateurs et de tâches.

10.6 – Multitraitement.

Après la multiprogrammation un nombre croissant de SE a offert le multitraitement (multiprocessing). Le multitraitement comporte deux aspects très différents:

- le cas où un seul SE gère plusieurs processeurs (cas classique des multiprocesseurs),
- le cas où plusieurs copies du SE, ou plusieurs SE différents dialoguent, chacun gérant son processeur, les ordinateurs des différents processeurs étant reliés entre eux (cas des multisystèmes, cf. tome I). Une même application peut voir ses différents processus tourner simultanément sous différentes copies du SE.

Une définition plus générale pourrait être: "Aptitude d'un SE à supporter plusieurs processeurs" [(WOL81)], mais cette généralité recouvre des réalités très différentes vis-à-vis des SE.

Certains SE ont été conçus, dès le départ, pour supporter des multiprocesseurs: OS1100, GCOS3, Multics par exemple. D'autres ont été modifiés afin de supporter les multiprocesseurs: Primos lors de l'introduction du biprocesseur 850, VMS lors de l'annonce du VAX-11/782; ils offrent alors des formes dégradées de gestion des multiprocesseurs comme nous l'avons souligné dans le tome I chapitre 17. Dans le monde des 370 d'IBM, seuls MVS et VM supportent les multiprocesseurs.

De nombreux systèmes plus récents et basés sur des multimicros (cas de Stratus, de Coralis 80 avec Cortex, etc.) ou des multiminis (cas de Tandem avec le SE Guardian) se présentent avec une copie du SE par processeur. Une classification plus fine fait apparaître une plus ou moins grande partie du SE dupliquée sur chaque processeur. Dans le cas des multisystèmes avec processeurs reliés par réseau local (bus en général), deux grandes tendances se dégagent:

- celle dans laquelle seul un noyau **minimum** est dupliqué, l'ensemble des services du SE étant complété par des processus distribués sur l'ensemble des processeurs: cas de Cortex, de ACK;
- celle de la duplication **totale** où chaque processeur est autonome et fournit tous les services du SE: cas de Tandem.

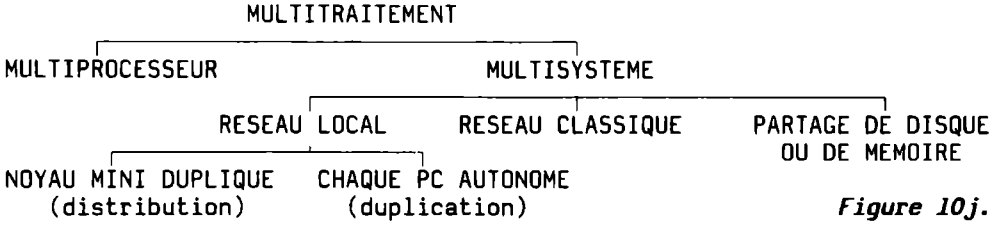


Figure 10j.

D'une façon moins "serrée" (loosely coupled, liaison CTC) des configurations de plusieurs systèmes (MVS avec JES, GCOS3 avec SMS) offrent un multitraitement mettant en commun la file des travaux par partage de disque.

Si les multiprocesseurs apportent une augmentation de la puissance globale, ils ne s'appliquent pas obligatoirement à toutes les applications: un transactionnel comme TDS sous GCOS3 ne pourra pas en bénéficier puisque GCOS3 n'a pas la notion de "multitasking" généralisé; GCOS3 ne voit pas chacune des transactions, il ne voit que TDS à qui il donne en temps convenu le contrôle d'un processeur; TDS à son tour redistribue ce processeur entre ses différentes tâches, mais il ne peut distribuer les autres processeurs que GCOS3 ne lui a pas attribués. Des essais rendant visibles de GCOS3 les tâches de TDS ont été réalisés. L'option MRO de CICS a une origine analogue.

Une analogie courante fait appel à un secrétariat avec une ou plusieurs secrétaires: une seule secrétaire fera la frappe, répondra au téléphone, classera le courrier, etc., elle fera de la multiprogrammation, interrompant une tâche pour une autre avant d'y revenir. Plusieurs secrétaires correspondront au multitraitement, chacune réalisant une seule tâche, ou un nombre réduit de tâches.

10.7 – SE multidimensionnel.

Le terme de "multidimensionnel" est utilisé pour traduire le fait que le SE accepte des requêtes soumises sous des formes différentes: travaux soumis de façon "batch", sessions interactives, activités locales et distantes. Ces formes sont détaillées ci-après, excepté le traitement par lots qui a fait l'objet d'une présentation en début du tome.

10.7.1 – Traitement par lots à distance (RJE).

Cette dimension est une extension du traitement par lots provoquée par l'apparition des liaisons télé-informatiques (remote batch processing, Remote Job Entry pour RJE). La lecture des travaux et l'édition des résultats est réalisée par JES sur MVS, par RGIN (Remote GIN) sur GCOS3.

10.7.2 – Temps partagé (time–sharing).

Le temps partagé représente la première dimension interactive apparue sur les ordinateurs. Elle est née avec l'utilisation des ordinateurs pour le calcul scientifique et avec les mini-ordinateurs.

L'utilisateur du temps partagé est généralement un informaticien. Il a accès à toutes les fonctions du système au moyen de commandes qui sont équivalentes au langage de commande (JCL) disponible dans le traitement par lot; IBM parlera de "transactions" pour les commandes exécutées sous TSO. Sur certains systèmes (OS1100, SE de CDC) le langage de commande est identique en traitement par lots et en temps partagé.

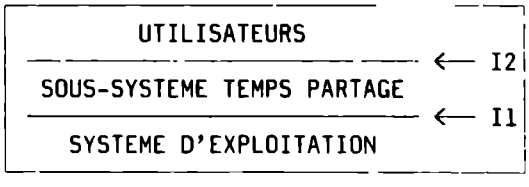


Figure 10k: dans un environnement temps partagé les deux interfaces I1 et I2 sont fonctionnellement équivalentes. Elles peuvent même être syntaxiquement identiques.

Des exemples de sous-systèmes temps partagé sont:
- TSS (Time Sharing Subsystem) sur DPS8 où une tâche maître (TSE pour Time Sharing Executive) joue un rôle de "mini SE" dans le monde du TSS. C'est elle qui crée le "multitasking" à l'intérieur du travail TSS, seul connu de GCOS3;
- IOF sur DPS7;
- TSO (Time Sharing Option) sur MVS, apparu en 1971 sur OS/MVT. Chaque utilisateur y dispose d'un espace d'adresses, tout comme une étape dans le traitement par lot. Sous MVT plusieurs utilisateurs partageaient une même région (TSO pouvait cependant contenir plusieurs de ces régions multi-utilisateurs). Après avoir eu une structure analogue à celle d'IMS (région de contrôle) il est maintenant intégré à MVS. 70 % des gros sites IBM en disposaient en 1983 selon une étude d'IDC;

- ICCF sous DOS/VSE;
- VOLLIE d'ADR sous DOS et DOS/VS, etc.

Il est possible de communiquer entre les mondes du temps partagé et du traitement par lots:
- envoi d'un travail depuis le temps partagé: commande SUBMIT de TSO, JRN de TSS;
- consultation des sorties (SYSOUT) depuis le terminal interactif: commande JSTS SYSOUT de TSS;
- interrogation de l'état du travail: commande JSTS de TSS;
- suppression (cancel, abort) du travail: commande JABT de TSS.

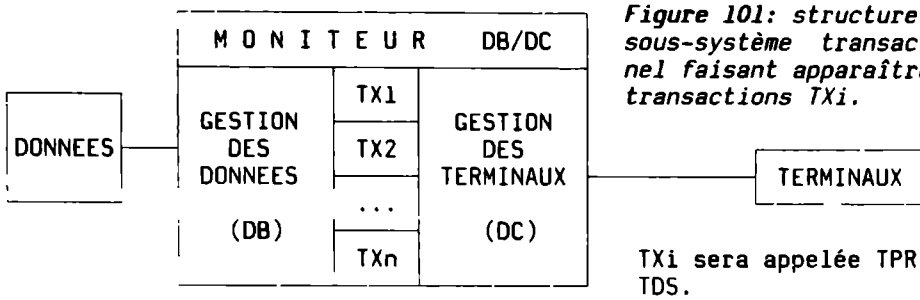


Figure 101: structure d'un sous-système transactionnel faisant apparaître les transactions TXi.

TXi sera appelée TPR dans TDS.

10.7.3 – Transactionnel (transactional).

Ce mode constitue le deuxième mode interactif apparu sur les ordinateurs: contrairement au temps réel au sens strict, il a été suscité par le monde de la gestion. Pour l'informaticien une transaction est une application écrite dans un environnement particulier, différent du traitement par lots puisqu'il dialogue avec des terminaux, mais elle n'est qu'un cas particulier de programme. Pour l'utilisateur la transaction est une opération qu'il effectue dans le cadre de son travail et qui est matérialisée par l'échange de messages avec l'ordinateur, avec le serveur.

10.7.3.1 – Cas général.

Un sous-système transactionnel est un ensemble fermé d'applications (appelées transactions): l'utilisateur est généralement un non-informaticien et répond aux questions posées par les transactions. Contrairement au temps partagé, il n'a pas accès directement aux fonctions du système: il ne peut pas frapper des commandes du SE correspondant aux interfaces I1 et I2 de la figure 10k. Il ne peut rien faire qui n'ait été prévu dans le sous-système, il ne peut pas créer de nouveaux objets (programmes, fichiers). L'ajout de nouveaux objets est fait par des spécialistes à l'aide d'une génération du sous-système au cours de laquelle on décrit les transactions, les terminaux, les fichiers, les paramètres d'exécution.

La figure 101 résume la structure générale d'un sous-système transactionnel, parfois appelé logiciel DB/DC (Data Base/Data Communication) puisqu'il gère des bases de données et des terminaux

reliés par des lignes de communication). Les sous-systèmes transactionnels les plus connus sont:

- TDS (Transactional Driven System) sur DPS7 et DPS8 où les transactions (les programmes transactionnels) s'appellent des TPR (Transactional Processing Routine). Sur DPS7 chaque transaction constitue un processus et profite donc du "multitasking" généralisé du SE ainsi que des protections apportées par la segmentation de l'architecture du DPS7. Sur DPS8 avec GCOS3 le moniteur ("l'exécutive" de TDS) est seul à être connu du SE et il crée dans sa partition un système multitâches local qui bénéficie de mécanismes apportés par l'architecture DPS8: les transactions font appel à l'exécutive de TDS par une instruction de type SVC (appel superviseur, instructions appelées MME, DeRaiL dans les DPS8); chaque transaction est isolée des autres sur le plan de l'accès à la mémoire et une transaction défectueuse ne viendra pas perturber les autres. La commutation de processus y est estimée plus performante que celle, trop générale, réalisée par le distributeur;

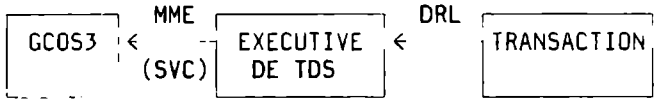


Figure 10m: les différents niveaux de contrôle avec TDS et GCOS3.

- TPS (Transaction Processing System) sur DPS8 et Mini6. Sur DPS8, le moniteur s'appelle TPE (TP Executive) et il gère des transactions appelées TPAP (TP APplications), chaque TPAP est un travail au sens de GCOS3, structure proche de celle d'IMS;
- DTF (Distributed Transactional Facility) sur Mini6. Nous reviendrons sur ce terme de distribué par la suite;
- TMF (Transaction Monitoring Facility) de Tandem;
- CICS sous DOS et OS d'IBM a débuté en 1968. Toutes les transactions y partagent le même espace d'adresses (16 Mo avant l'architecture étendue XA) et il n'y a donc pas de protection entre les différentes transactions: une transaction peut très bien écraser le code ou les données d'autres transactions, du noyau de CICS même. L'option MRO (Multiple Region Option en 1979) étend l'espace adressable à plusieurs AS, mais il faut pouvoir couper la base de données en plusieurs bases indépendantes, deux AS de MRO ne pouvant partager la même base; les transactions sont elles aussi associées de façon fixe à un AS. [YEL85] indique qu'un 3090-200 chargé à 70 % avec des applications Cobol sur des fichiers VSAM atteint 80 transactions par seconde, 120 avec l'option MRO qui permet d'atteindre la saturation du processeur. Comme TSO, CICS connaît les utilisateurs par leur identification (ID) et non pas les terminaux physiques comme IMS. Il y a 25000 licences CICS dans le monde selon le Gartner Group dans [KAD86]. D'après une enquête de Xephon, menée principalement en Angleterre et portant sur 121 sites, 69 % d'entre eux déroulent entre 10000 à 100000 transactions par jour. La figure 10n présente, dans le cas de CICS, une structure plus détaillée que la structure générale de la figure 10l;

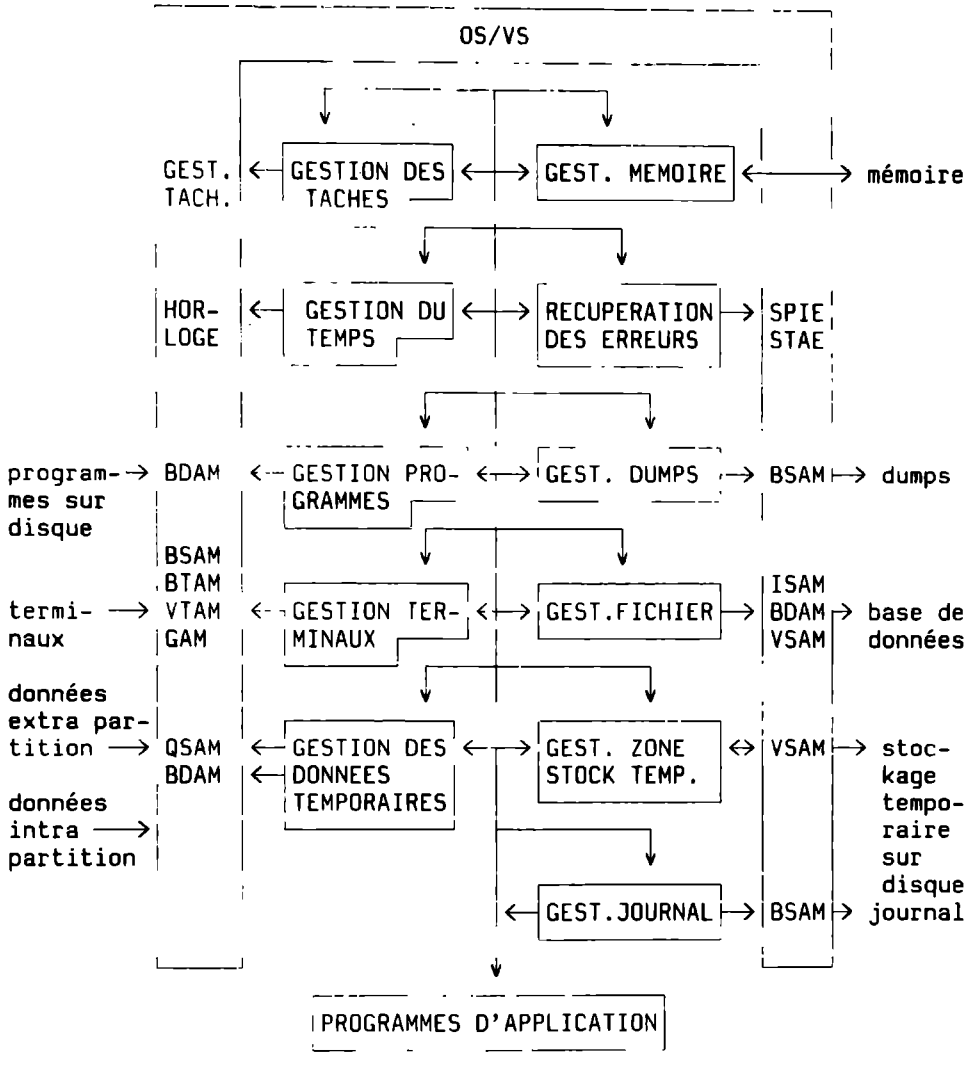


Figure 10n: structure de CICS selon [IBM74X].

- IMS sous MVS d'IBM: chaque transaction (MPP ou MPR) y dispose d'un espace d'adresses au sens de l'architecture des 370, ce qui permet de tirer aisément profit des multiprocesseurs. Si CICS convient à de petits réseaux, IMS est plutôt utilisé sur de grosses machines, avec un grand nombre de terminaux et de transactions. Contrairement à CICS, il connaît des terminaux physiques identifiés par un LTERM (Logical TERMinal) au moment de sa génération. Quand il reçoit un message, il le met en file d'attente, le traite dès que possible

puis met la réponse à ce message en file d'attente, jusqu'à ce qu'il soit en mesure de l'envoyer au terminal, et non pas à l'utilisateur, qui a émis le message. Autre différence avec CICS, IMS a ainsi un fonctionnement **asynchrone**: un message émis ne bloque pas le terminal émetteur. Lorsque l'on veut synchroniser plusieurs transactions, enchaîner plusieurs échanges, on parle dans IMS de transactions **conversationnelles**.

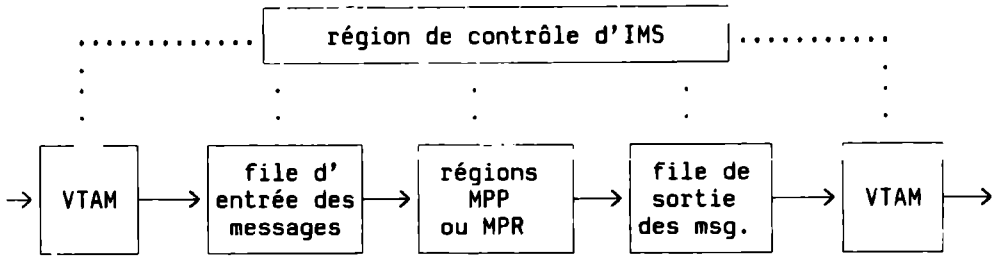


Figure 10o: structure globale d'IMS faisant apparaître les files de message.

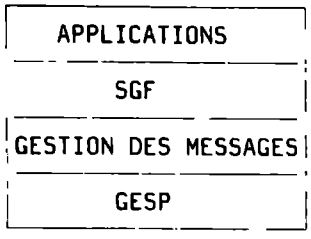
L'exécutive d'IMS (région de contrôle) choisit la région (MPR) où exécuter chacune des transactions: ceci correspond à son activité de planification (scheduling). La MPR appelle ensuite la copie du programme (la transaction) qui traite le type de message reçu. Afin d'optimiser le fonctionnement, il est possible de ne pas dérouler cette activité de planification/allocation pour chaque message: une transaction peut ainsi rester dans la même région pour traiter plusieurs messages consécutifs du même type. Pour éviter les effets secondaires, à savoir une monopolisation des régions MPR par quelques transactions, une limite (PLMCT, Process LiMit CounT) est fixée au nombre de messages consécutifs ainsi traités.

L'optimisation du chargement du code se retrouve dans de nombreux sous-systèmes transactionnels. Les méthodes les plus courantes sont:

* l'antémémoire réalisée par conservation dynamique en mémoire des modules les plus souvent utilisés;
* chargement depuis une autre partition ou un autre espace d'adresses: cas d'IMS avec CMS;
* la duplication des modules de code sur plusieurs disques, le disque le plus disponible au moment de la requête étant utilisé;
* le chargement simultané depuis plusieurs disques en passant par des chemins différents (IMS);
* le préchargement (preload), en mémoire virtuelle et/ou en mémoire centrale, au risque d'occuper de l'espace avec des modules peu sollicités. IMS permet ainsi un préchargement dans la zone privée de l'AS, ou dans la LPA (option "fast path").

Notons que dans MVS il n'y a pas partage du code applicatif entre deux espaces d'adresses; si la même transaction est chargée dans plusieurs espaces d'adresses, elle figurera sous forme de plusieurs copies. Dans

CICS les transactions sont réentrantes, dans TDS de GCOS3 elles sont réutilisables en série (cf. tome V).



Tous les périphériques sont vus comme des fichiers, les processus aussi d'ailleurs.

Figure 10p: structure du logiciel sur Tandem.

Les sous-systèmes transactionnels listés ci-dessus proviennent des constructeurs d'ordinateurs associés. Il existe aussi des sous-systèmes transactionnels fournis par des sociétés de service (third party).

Des constructeurs récents se sont lancés sur ce marché du transactionnel avec des systèmes nouveaux qui sont généralement classés dans les systèmes à tolérance de panne (cf. tome I). Ces systèmes offrent une disponibilité très élevée; le plus ancien est Tandem qui comptait en 1984 plus de 2000 systèmes installés et utilise l'expression "non stop" pour ses machines afin de bien souligner l'aspect disponibilité. Un des noms du SE Guardian de Tandem était d'ailleurs initialement T/TOS pour Transactional OS (cf. [TAN75]). La figure 10p présente la structure en couches du SE de Tandem, le rôle de la gestion des messages qui banalise acteurs (processus) et ressources (appareils, fichiers) y est mis en évidence, insistant sur l'aspect distribué et donc résistant des applications.

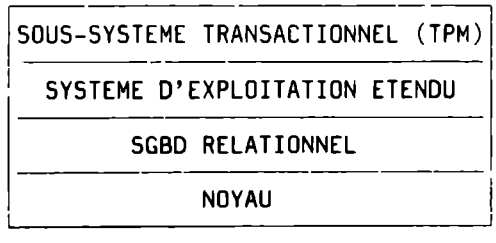


Figure 10q: couches de logiciel dans la machine de Synapse.

TPM: gestion des sessions, E/S écran, reprises.
SE étendu: gestion des noms, gestion des E/S logiques.
SGBD R: gestion de l'intégrité, accès à la base de données, accès au dictionnaire des applications.
NOYAU: gestion des processus, gestion de la mémoire, gestion des reconfigurations.

Synapse qui se situe dans ce marché présente son logiciel avec une structure en couches illustrée figure 10q. On y constate l'importance du SGBD implémenté directement au-dessus du noyau: situation typique du transactionnel, différence que nous avons déjà mentionnée par rapport au temps partagé.

Ajoutons pour clore cette liste des sous-systèmes transactionnels, ACP (TPF maintenant) d'IBM, vieil SE spécialisé pour

le transactionnel, ou le SE spécial TOP développé sur 1100 par Air France dans le même but qu'ACP, la réservation de place.

Depuis 1981 se développent des applications **télématiques**. Elles se déroulent généralement dans un environnement de type transactionnel et se caractérisent par des connexions fréquentes mais courtes, avec des interactions lentes qui consomment relativement peu de ressources. Les SE dans lesquels les coûts de connexion, d'établissement de session, sont élevés, ont dû réagir et fournir de nouveaux sous-systèmes comme GTM (Generalized Transactional Monitor) chez IBM France.

10.7.3.2 – Transactionnel et temps partagé.

Sous-système transactionnel et temps partagé posent des problèmes très différents sur le plan de leur gestion par le système d'exploitation: dans un sous-système transactionnel le code est associé à des types de message et non pas à des utilisateurs, il y aura ainsi n transactions exécutables simultanément pour N, très supérieur à n, utilisateurs actifs, n étant choisi en fonction de chaque installation. Le taux de multiprogrammation ou de "multitasking" y est donc réglable a priori.

Dans un sous-système temps partagé il faut maintenir une activité (un processus) par utilisateur actif car les utilisateurs ne font pas appel a priori aux mêmes codes; au niveau de la gestion mémoire (cf. chapitre 11) il faudra des mécanismes de vidage (swapping) mis en oeuvre si nécessaire dès que l'utilisateur a reçu une réponse du système, son temps de réaction étant lent vis-à-vis de ce qui se passe à l'intérieur de l'ordinateur. Du moins en théorie.

Une autre différence entre temps partagé et transactionnel est que l'utilisateur du transactionnel a absolument besoin d'accéder, de partager des données; l'utilisateur du temps partagé est plus autonome dans ses besoins et un micro-ordinateur doté d'un bon logiciel de communication avec l'ordinateur central lui conviendra mieux (cf. [PIN86]).

10.7.3.3 – Transactionnel distribué.

Avec l'arrivée des réseaux d'ordinateurs, apparaissent des applications distribuées qui tournent dans les sous-systèmes transactionnels de plusieurs ordinateurs. Une application sera dite distribuée lorsqu'elle met en oeuvre plusieurs environnements informatiques géographiquement différents. L'environnement transactionnel d'une machine Tandem suppose obligatoirement plusieurs processeurs avec chacun leur mémoire et leur copie du système d'exploitation, mais au niveau d'une seule machine Tandem on ne parlera pas de transactionnel distribué. On trouve encore les termes de:

- transactionnel à plusieurs niveaux,
- transactionnel distribué,
- transactionnel coopératif.

Ainsi Bull propose des transactions distribuées entre GCOS6/DTF et DPS8/TDS. Chez IBM, les produits comme ISC et MSC sont destinés à la

réalisation d'applications distribuées.

[LEE84] définit un système distribué comme un système dans lequel "les programmes d'application et/ou les données résident dans des sites interconnectés séparés, et sont conçus de façon intégrée avec un contrôle étroit (tight)". Il classe les applications correspondantes en quatre classes selon que le traitement et le stockage se font sur le même site ou sur un site différent par rapport à l'utilisateur final:

CLASSE D'APPLICATION	SITE DE TRAITEMENT	SITE DE STOCKAGE CORRESPONDANT
SS	MEME	MEME
DS	DIFFERENT	MEME
SD	MEME	DIFFERENT
DD	DIFFERENT	DIFFERENT

Il compare ces transactionnels distribués selon cinq critères résumés dans le tableau ci-dessous:

- il y a transparence de lieu quand l'utilisateur n'a pas à savoir où se trouvent les transactions ou les données, il peut appeler les transactions depuis tout système interconnecté;
- la transparence de duplication (replication) des données concerne la possibilité de dupliquer une partie de la base de données, sans que l'utilisateur ait à gérer cette duplication;
- la transparence de concurrence concerne le contrôle des mises à jour simultanées, contrôle réalisé par le sous-système pour qu'il y ait transparence;
- la transparence des pannes permet d'assurer que l'ensemble des opérations déroulées sur chaque site a globalement bien fonctionné;
- la notion de transaction existe lorsque l'ensemble des opérations déroulées sur différents sites est considéré comme une unité de reprise:

TRANSACTIONNEL DISTRIBUE	TRANSPARENCE DE				NOTION DE TRANSACTION
	LIEU	DUPLICATION	CONCURRENCE	PANNE	
ENCOMPASS	*	(2)	*	*	
INGRES	*	*			
IMS/MSC	*(1)		*	*	*
CICS/ISC	*			*	*

(1) de façon limitée: un programme ne peut accéder qu'à la base locale
(2) il y a possibilité de duplication mais non transparence

10.7.4 – Télétraitement.

Avec l'accès à distance (remote), les systèmes d'exploitation ont abordé le traitement à distance (remote processing) appelé "télétraitement". Le télétraitement recouvre un certain nombre de

modes de fonctionnement assez différents selon que l'on accède au traitement par lots ou au traitement interactif. Les caractéristiques du télétraitement sont les suivantes:

1. les terminaux sont reliés à l'ordinateur par des lignes de communication fournies par les PTT;
2. les entrées ont un caractère aléatoire: les utilisateurs envoient leurs réponses/questions quand cela leur convient;
3. dans le cas de l'interactif, le traitement doit être immédiat. En interactif, chaque requête est prise en compte individuellement et la réponse est fournie immédiatement. En traitement par lots, le système collecte les entrées, les traite quand il l'estime convenable, et le traitement se fait par quantités et non plus au niveau du message;
4. le temps de réponse doit être court. En traitement par lots, on optimise la charge des ressources, en interactif, on optimise le temps de réponse;
5. le système est utilisé simultanément par de nombreux utilisateurs qui peuvent partager un même programme.

Notons que le terme de télétraitement tombe peu à peu en désuétude et est repris avec un sens différent par "télématique".

	MODE DE FONCTIONNEMENT	
	DIFFERE	DIALOGUE
LOCAL	L	TP TR
DISTANT	LD TI	TP TR TI TM

L : traitement par lots
LD: traitement par lots à distance
TP: temps partagé
TR: transactionnel
TI: télé-informatique
TM: télématique

10.7.5 – Temps réel (real time).

Nous avons déjà abordé cette expression de temps réel au sujet des systèmes d'exploitation qui ont cette vocation. Cette expression est aussi très utilisée dans le contexte du transactionnel et du télétraitement. Pour les puristes, le transactionnel se distingue du temps réel par l'échelle temps: le transactionnel comme l'interactif (mode dialogué) en général relève des interactions avec les utilisateurs humains alors que le temps réel oeuvre dans le domaine des équipements qu'il n'est pas question de faire attendre démesurément. Le télétraitement est une notion orthogonale par rapport au temps réel: il peut y avoir temps réel sans télétraitement si les équipements pilotés sont locaux, c'est-à-dire se trouvent dans la même pièce, dans le même bâtiment que l'ordinateur, s'il n'est pas fait appel aux fonctions "réseau" classiques que nous allons introduire dans le chapitre 13. Il y aura simultanément temps réel et télétraitement s'il est fait appel à des fonctions réseau pour atteindre les équipements en question. Un effort de standardisation française d'un noyau temps réel a conduit à SCEPTRE (cf. [BRO84]).

10.8 – Mémoire virtuelle.

Utilisée dès 1962, sur l'ATLAS de l'université de Manchester et de la société Ferranti, la mémoire virtuelle offre une nouvelle technique de gestion des différents niveaux de la hiérarchie de mémoires dont dispose un ordinateur. Elle vient s'ajouter aux techniques d'antémémoire et de mémoires associatives évoquées dans le tome I. Comme très fréquemment en informatique elle est devenue universellement adoptée depuis qu'IBM l'a introduite avec la série 370. Quelques SE vivent encore au début des années 80 sans mémoire virtuelle de type IBM: GCOS3, OS1100, COS de Cray, par exemple.

[CAS78] justifie la mémoire virtuelle introduite dans les 370 par les huit points suivants:

1. les problèmes posés par le vidage (roll-in, roll-out),
2. la fragmentation externe de la mémoire centrale,
3. la difficulté pour développer des applications avec les techniques de recouvrement (overlay),
4. l'avantage de l'adaptation dynamique à la taille mémoire réellement disponible,
5. la compatibilité réalisée entre les grandes et les petites tailles de mémoire,
6. le besoin de protection et de partage (limitation des 15 clés du 360),
7. les avantages apportés par les entrées/sorties virtuelles,
8. la réalisation de machines virtuelles.

Les deux techniques de mémoire virtuelle (MV) et d'antémémoire (AM) ont en commun leurs aspects **dynamiques**: c'est sur demande instantanée (non planifiée) que le bloc d'information est amené dans le niveau inférieur. La mémoire centrale n'est plus qu'une antémémoire vis-à-vis de la mémoire virtuelle, l'ensemble du mécanisme étant transparent pour l'utilisateur.

Nous verrons plus loin que les deux grandes techniques liées à la MV sont la pagination et la segmentation. Bien que certains, comme Burroughs, l'utilisaient depuis longtemps, la MV fut introduite par de nombreux constructeurs dans les années 70. Si elle a étendu les fonctionnalités des ordinateurs, elle provoque parfois des déperditions (overhead) beaucoup plus élevées: les ressources processeur nécessaires pour supporter une charge interactive donnée ont doublé entre OS/MVT (SE à mémoire réelle) et son successeur à mémoire virtuelle OS/VS2 version 2.

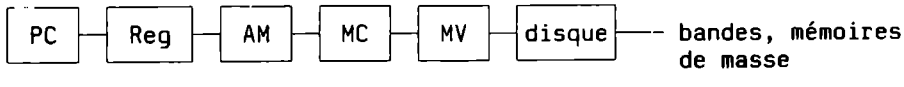


Figure 10r: rappel de la hiérarchie des mémoires.

10.9 – Machines virtuelles.

Le terme de machine virtuelle est utilisé dans de nombreux contextes, dès qu'il y a plusieurs utilisateurs sur un ordinateur

(chacun voit "sa" machine qui n'est pas tout à fait la machine réelle) ou dès qu'il y a visibilité d'une machine plus évoluée que celle offerte par le seul matériel: l'ensemble matériel plus SE est parfois appelé machine virtuelle. A la limite la machine vue par l'utilisateur peut être très différente de ce qu'offre le matériel réellement utilisé, plusieurs utilisateurs voyant des machines différentes.

Dans cette section nous nous limiterons aux réalisations les plus classiques. Un hyperviseur y joue vis-à-vis des SE (des machines virtuelles) un rôle analogue à celui joué par un SE vis-à-vis des processus. Il ne gère (multiplexe) cependant que les ressources physiques.

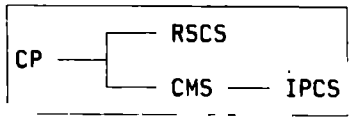
Le cas le plus connu est l'implémentation faite par IBM sous le nom de VM/370 (Virtual Machine annoncé en 1972) puis VM/XA. Il a démarré en tant que projet au début des années 60 sous le nom de CP, les premiers systèmes commercialisés s'appelant CP-67. CP-67 fut annoncé en 1965 sur un 360/67, un 360/65 modifié. Notons que CP fut développé en parallèle (et en concurrence) avec Multics. Contrairement à Multics qui avait pour objectif initial le partage contrôlé des informations entre les utilisateurs, CP établissait des frontières étanches entre les utilisateurs: les machines virtuelles ne pouvaient communiquer que par les périphériques, lecteurs/perforateurs de cartes virtuels, minidisques partageables (partie d'un disque qui est affectable à une machine virtuelle, sorte de volume logique au sens du chapitre 7). CMS (Conversational Monitor System), un SE purement interactif, fut simultanément développé. Notons que dans le cas de VM les machines virtuelles supportées font toutes partie d'une même architecture externe, celle de la gamme 370, c'est-à-dire celle de la machine réelle. [VM79A] définit les machines réelles et virtuelles de la façon suivante:

MACHINE REELLE	MACHINE VIRTUELLE
.un processeur réel	.un processeur virtuel
.une mémoire principale réelle	.une mémoire principale virtuelle
.des lecteurs de cartes et	.des lecteurs de carte et
.des imprimantes réelles	.des imprimantes virtuelles
.des disques et bandes réels	.des disques et bandes virtuels

VM/370 se présente sous forme de quatre composantes:

- l'hyperviseur CP (Control Program), "Superviseur de la machine réelle. Il gère les ressources de la machine réelle de telle sorte qu'un certain nombre de machines virtuelles semblent exister. Il supporte plusieurs mémoires virtuelles, chacune d'elles étant la mémoire principale (imaginaire) d'une machine virtuelle. En outre CP est capable de supporter des unités d'E/S «imaginaires»." ([VM79A]);
- CMS, un SE interactif qui ne peut fonctionner hors VM;
- RSCS (Remote Spooling Communications Subsystem). Il gère les spoules et permet l'échange de données avec des stations éloignées, entre les machines virtuelles. Il fait partie de l'architecture NJE dont VNET est une réalisation. Comme CMS il ne peut fonctionner hors VM;
- IPCS un outil de gestion des incidents destiné à la maintenance IBM.

Cette définition fait apparaître la notion de machine virtuelle comme une extension de la notion de mémoire virtuelle.



Composants de VM/370.

L'avantage de VM est:
- d'offrir un environnement interactif (CMS) plus convivial que celui de TSO, l'interactif de MVS;
- de faciliter les conversions entre SE, de DOS à MVS par exemple. Ceci était surtout un argument lorsque les utilisateurs n'avaient qu'un seul ordinateur. Beaucoup ont maintenant une machine de test, de développement, en plus de la machine de production et cet avantage de VM a quelque peu diminué;
- de tester de nouvelles versions du SE.

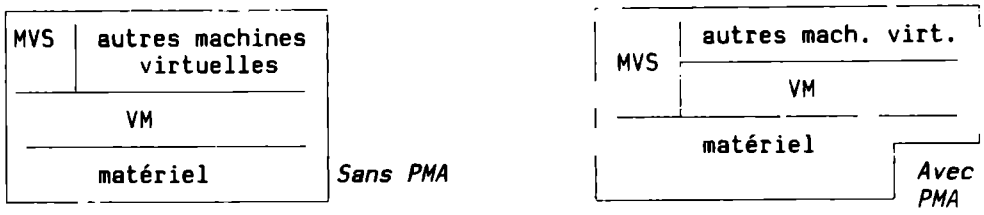
[VM79A] ajoute comme domaines d'application de VM:
- la mise au point de programmes dans une machine virtuelle de test. On ne risque pas de "planter" la machine de production;
- la formation des pupitreurs.

L'addition d'un hyperviseur n'améliore pas les performances, bien au contraire, puisqu'il ajoute un second niveau de gestion à celui déjà assuré par les superviseurs des SE. Ainsi le temps processeur est d'abord alloué par l'hyperviseur à un SE (une machine virtuelle), puis ce SE répartit ce temps entre ses différents processus; la mémoire est gérée et par l'hyperviseur et par les gestionnaires de mémoire centrale (GMC) de chaque SE: [BOZ84] mentionne que l'algorithme de gestion de la mémoire dynamique de VM/SP consomme de 11 à 20 % du temps processeur passé en mode superviseur. Les tables de page (chapitre 11) du SE ne servent à rien, mais CP doit les mettre à jour. La déperdition globale est de 30 à 40 % [DUB86].

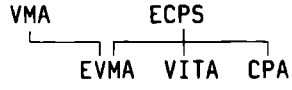
Dans le cas de VM/370, CP s'exécute en mode privilégié (superviseur dans la terminologie IBM), les SE sont de ce fait restreints aux instructions non privilégiées et toute exécution d'une instruction privilégiée provoque une interruption traitée par CP. Ceci constitue aussi une source de ralentissement que de nombreuses "aides microprogrammées" (assist) tentent de limiter, [SCH85] indique que 80 % de la déperdition (overhead) de MVS sous VM en mode V=V (voir chapitre 11) sur un processeur "non assisté" est due à la réimplantation des pages quand les AS changent et à la simulation des instructions privilégiées comme SIO. [SCH85] donne la performance relative des SE lorsqu'ils tournent sous VM:
0,82 à 0,92 pour les machines en V=R et avec ECPS complet,
0,75 à 0,85 pour celles avec VMA seulement,
0,1 à 0,4 pour celles sans "assistance" aucune,
0,9 à 0,99 pour celles avec PMA, avec canaux et périphériques dédiés.

L'option **PMA** (Preferred Machine Assist sur 370) consiste à faire disparaître VM de temps en temps, lorsque la "machine préférée" s'exécute; elle correspond à des microprogrammes et au logiciel

VM/HPO. A un instant donné, soit VM, soit MVS, la machine préférée, interface directement avec le matériel. PMA est une sorte d'hyperviseur d'hyperviseur ([DUB86]) qui dédie une partie de la mémoire centrale et des canaux à MVS. Cette option est à rapprocher de l'option MDF de Amdahl, bien que MDF offre plus de fonctionnalités: il gère des canaux en mode 370/XA et d'autres en mode 370.



Avec **VM/XA** CP sait donner la main à des machines XA ou 370, et on obtient les performances relatives suivantes:
0,7 à 0,8 avec MA (Migration Aid) PM (Preferred Machine),
0,5 avec MA en V=V,
0,9 à 0,92 avec SF (System Facilities) PXAG (Preferred XA Guest),
0,86 à 0,88 avec SF P370G (Preferred 370 Guest).



EVMA: Expanded VMA
VITA: VIrtual Timer Assist
CPA : CP Assist

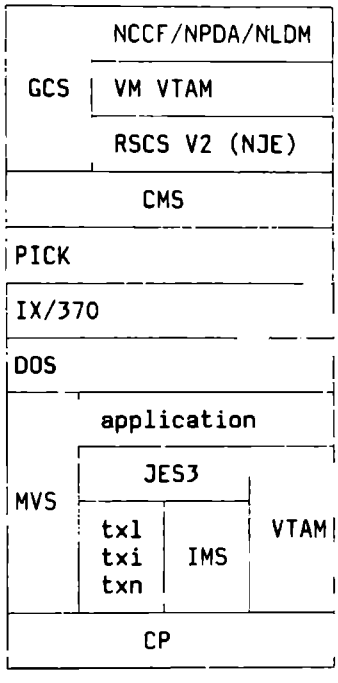
VMA (VM Assist) est un ensemble de microprogrammes évitant d'activer CP (le noyau de VM) dans un certain nombre de situations spécifiées dans le registre CR6: exécution de 7 instructions privilégiées 370 gérant les clés (IPK, SPKA), associées à la pagination (LRA, RRB), etc., exécution des SVC; VMA intervient aussi dans la gestion des tables cachées (shadow table). Selon [MKI79] l'effet de VMA est:

	370/135		370/145		370/158	
DEBIT RELATIF EN MODE LOT	DOS/VS	VS1	DOS/VS	VS1	DOS/VS	VS1
SANS VMA	0,34	0,26	0,48	0,35	0,37	0,21
AVEC VMA	0,66	0,75	0,79	0,70	0,69	0,50

ECPS:VM/370 est aussi un ensemble de microprogrammes qui s'ajoutent à VMA. Il comporte de nouvelles instructions (STECPSVM, celles de code opération E6) et correspond donc à une évolution architecturale des 370. La composante EVMA est appelée pour certaines des instructions simulées par VMA et simule totalement les instructions telles que PTLB, STPT, TCH, partiellement les instructions SCRC, SIO, SIOF, SPT.

Une machine virtuelle comme MVS ne sait pas qu'elle tourne sous VM, pour MVS tout se passe comme s'il n'y avait pas d'hyperviseur. Seules des raisons d'amélioration des performances peuvent conduire à rendre la machine virtuelle consciente de la présence de l'hyperviseur; c'est le cas de DOS et VS1 dont le mécanisme de dialo-

gue avec VM est appelé "handshaking", basé sur l'utilisation de l'instruction DIAGNOSE: contrairement à une SVC on appelle ainsi directement CP, ce qui évite les "program checks". Des redondances fonctionnelles sont ainsi évitées: sous VS1 une faute de page ne provoque plus l'arrêt de la machine virtuelle VS1, mais le seul arrêt de la tâche ou de la partition concernée. [MKI79] décrit une charge de type lot dont le débit relatif passe de 0,24 à 0,54 avec "handshaking", sans VMA, le débit de 1 correspondant au fonctionnement de VS1 hors VM. Avec VMA ces débits relatifs sont de 0,69 et 0,79 bien que la consommation de temps processeur total baisse de 20 %.



Exemple d'un ordinateur IBM fonctionnant sous VM avec plusieurs machines virtuelles:
.CMS pour l'interactif
.DOS le SE de la gamme moyenne
.MVS le SE du haut de gamme
.NCCF le logiciel d'administration réseau
.VM VTAM l'ouverture SNA pour VM
.RSCS le logiciel réseau et spoule de VM
.PICK un SE orienté SGBD relationnel
.IX/370 implémentation d'Unix sur 370

GCS implémente la notion de groupe de machines virtuelles dans un but d'optimisation.

txi: transaction

On trouve aussi cette notion d'hyperviseur sur le DPS88 de Bull avec GCOS8, pour gérer en parallèle GCOS8 et le système de test et de diagnostic. Dans le cas de GCOS3 "accomodé", Multics joue le rôle d'hyperviseur. Bull n'a jamais mis en avant les hyperviseurs de ses DPS7.

L'**iAPX386** a deux modes de fonctionnement dont l'un fait apparaître le terme de "machine virtuelle":

- le mode **réel** dans lequel il se comporte comme un 8086 accéléré pouvant manipuler des opérandes de 8 bits, 16 bits mais aussi 32 bits;
- le mode **virtuel** dans lequel sont disponibles tous les mécanismes de pagination, segmentation, commutation de processus. Dans ce mode peuvent être exécutés des processus qui se comportent comme s'ils s'exécutaient sur un 8086 lorsque le bit VM (Virtual Mode 8086) du

mot d'état EFLAGS est positionné. Il peut alors y avoir plusieurs "machines virtuelles" sur le 386, une pour chaque processus avec le bit VM à un.

10.10 – Systèmes d'exploitation distribués.

10.10.1 – Principes généraux.

Nous avons vu précédemment l'évolution des systèmes d'exploitation qui a conduit:
- à la multiprogrammation,
- aux possibilités multitâches,
- au support de plusieurs processeurs (multitraitement),
- avec des fonctions réparties "verticalement" sur plusieurs unités matérielles (processeur central, processeur d'E/S, processeurs de gestion de bases de données, périphériques intelligents, etc.).

Mais ces SE restent centralisés: ils contiennent un maître central qui dispose d'une représentation centralisée (dans la mémoire centrale) de **tous** les éléments de la configuration. Lorsque l'on évoque la distribution, quatre éléments peuvent être distribués:
- le matériel,
- les données,
- les applications,
- le contrôle comme les systèmes d'exploitation,

et pour certains tout système distribuant l'un de ces éléments est appelé système distribué.

La réalisation d'applications distribuées sur des processeurs géographiquement dispersés (au sein d'un réseau par exemple) et/ou sur des processeurs qui peuvent changer (en cas de panne, en cas de ré-équilibrage des charges par exemple) nécessite un contrôle distribué, donc des SE disposant de caractéristiques particulières. [KAI84] donne comme définition des systèmes **répartis** (nous ne ferons pas de différence ici entre réparti et distribué):
1. un système comportant un nombre quelconque de processus système et de processus utilisateur;
2. avec une architecture modulaire comprenant plusieurs processeurs, parfois en nombre variable;
3. les communications entre processus sont réalisées par transfert de message sur une voie de communication;
4. la responsabilité de certaines décisions n'est pas confiée à un processus unique mais est répartie entre plusieurs éléments;
5. les délais de transmission des messages sont variables si bien qu'il est impossible de fixer a priori des délais d'attente au terme desquels l'état du système est connu avec certitude.

Il y a SE réparti lorsque:
- les ressources ne sont pas attribuées a priori à des tâches fixes,
- les processus n'ont pas accès à une représentation unique et certaine de l'état du système.

L'étude de tels systèmes fait l'objet de nombreuses recherches et expériences dans le monde. En France ils ont été le sujet du projet

CHORUS à l'INRIA (cf. [BAN80], [GUI82], [ZIM81]). Ces études conduisent à des logiciels considérés comme un ensemble de processus qui:
- communiquent par **messages**, ainsi la façon de communiquer est la même, que les deux processus soient ou non dans le même processeur. [BAS82] oppose ainsi les SE orientés message aux SE orientés procédure comme Unix;
- s'identifient par **nom** et non pas par adresse: un processus aura le même nom quelque soit le processeur sur lequel il s'exécute;
- arment un temporisateur ("réveille-matin", time-out) chaque fois qu'un message est envoyé à un processus: si ce dernier ne répond pas au bout d'un certain temps, le processus émetteur en conclut qu'il n'est plus opérationnel, et il prend les mesures qui s'imposent.

[BAS82] avance que les SE orientés message offrent:
- une meilleure modularité et sont plus facilement extensibles puisqu'il suffit d'ajouter des processus, indépendants de ceux qui existent déjà. Il n'y a pas à modifier le noyau qui est généralement complexe;
- une meilleure isolation fonctionnelle grâce à l'espace mémoire indépendant que le matériel peut procurer à chaque processus comme nous le verrons dans le chapitre 11.

10.10.2 – Exemples.

On trouve de tels systèmes dans certains SE de la SM90 (machine conçue par le CNET et produite par de nombreux constructeurs français: SPS7 de Bull, CSEE, TRT, TELMAT, etc.); c'est aussi le cas du SE ATHOS des X83 de THOMTIT qui sont, entre autres, des commutateurs de paquets (commutateurs de transit dans TRANSPAC 2G). Les SE des nouveaux autocommutateurs téléphoniques (E10 de CIT-Alcatel, System 12 d'ITT) sont bâtis sur des concepts analogues.

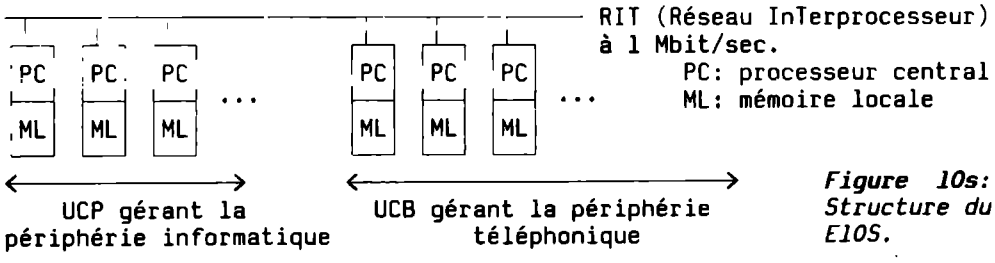


Figure 10s: Structure du E10S.

La notion de "Machine Logique" dans le E10S (cf. [BAL81], [MAI81]) correspond à un sous-ensemble fonctionnel cohérent de programmes regroupés à la génération et pouvant être chargés dans toute machine physique; une machine logique est référencée par son nom, indépendamment de la machine physique sur laquelle elle a été chargée; elle constitue une unité d'édition de liens et de chargement. Les machines logiques sont composées de processus asynchrones qui communiquent par messages physiquement échangés par les processeurs, il n'y a pas de mémoire commune sur le E10S. Le processus est à la machine logique ce qu'est un sous-programme vis-à-vis d'un programme. Le

"logiciel de base" est chargé dans chaque processeur physique et comprend une séquence d'initialisation en REPROM (SIPROM) ainsi que le SGTE (Système de Gestion des Tâches et des Echanges). Le SGF constitue une machine logique (il n'est donc pas inclus dans le noyau) et ne figure qu'à un seul exemplaire dans le système, d'autres machines logiques gèrent les périphériques informatiques.

L'autocommutateur MT25 (cf. [LAG81]) a un logiciel décomposé en "sous-ensembles fonctionnels" (SEF) qui dialoguent aussi par messages. Le SE comprend deux SEF: le SOP (Système OPératoire) qui gère les processus, le MUC (Matériel Unité de Commande) qui est plus orienté vers la gestion des ressources périphériques, vers les problèmes de maintenance.

[PHI86A] décrit un système distribué dont le SE s'appelle **DRM** (Distributed Real time Multiprocessor); les applications y sont écrites sous forme de "machines logicielles" (SOMA, SOftware MAchine), un SOMA étant un ensemble de processus chargés dans une "machine matérielle" (HAMA, HArdware MAchine). Les SOMA dialoguent de façon asynchrone par messages et boîtes aux lettres (BàL), un SOMA peut avoir plusieurs BàL, chacune étant associée à un certain type de message. DRM utilise aussi des "tubes" (pipe) pour les communications synchrones entre SOMA. Citons encore Auros du System 4000 d'Auragen ([VIL84]), une évolution d'Unix avec une communication par messages ajoutée dans le noyau afin d'offrir un multisystème tolérant les pannes adapté aux besoins des applications transactionnelles. De même ROS de Ridge (le SPS9 de Bull) présente une visibilité Unix mais est basé sur la communication par message entre processus.

Certaines architectures de microprocesseur sont basées sur ces considérations: l'iAPX432 d'INTEL par exemple.

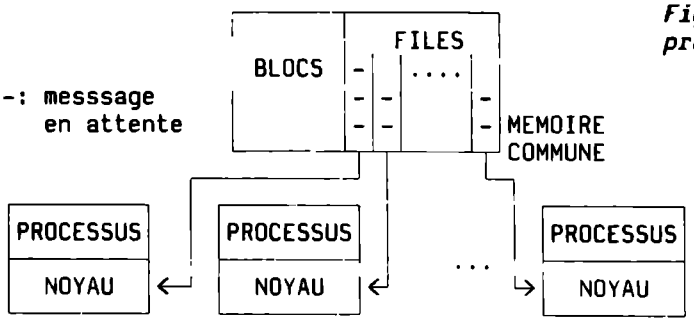


Figure 10t: gestion des processus par Cortex.

Cortex sur Coralis 80 de CSEE s'inspire de la philosophie de CHORUS. Un noyau très restreint est dupliqué dans chacune des mémoires locales des processeurs (Z80) qui constituent un Coralis 80. Le distributeur de chaque copie du noyau soit exécute un processus, soit déroule du code lié à une interruption, soit scrute sa file d'attente qui se trouve en mémoire commune. Une file d'attente contient des messages destinés aux processus qui résident dans la mémoire locale du processeur associé (cf. figure 10t). Chaque message provoque le lance-

ment du processus destinataire, processus qui ne peut pas être interrompu par un autre processus (fonctionnement multitâche en "série" au niveau d'un processeur). Seules les interruptions peuvent interrompre temporairement le processus en cours d'exécution, une séquence de traitement d'interruption ne peut pas être interrompue, d'ailleurs il n'y a qu'un seul niveau d'interruption.

Les processus communiquent donc par messages construits dans les blocs des différents pools de tampons de la mémoire commune. Chaque pool est protégé par un verrou grâce à une fonction de "test and set" (cf. tome I) simulée par la mémoire commune puisque le Z80 en est dépourvu au niveau du jeu d'instructions. La taille des tampons des différents pools va de 64 octets à 4 Ko (valeur limite fixée par la conception du matériel), la taille de chaque pool est fixe: il n'y a pas d'équilibrage dynamique en cours d'exécution en fonction des demandes réelles; tout est défini lors de la génération. La gestion de la mémoire commune est donc répartie entre les différentes copies du noyau dont la taille est très petite, environ 4 Ko. Un processus peut bien entendu se réactiver lui-même en s'envoyant un message. L'adressage des processus est physique et non pas basé sur les noms des processus: un processus émetteur doit savoir sur quel processeur s'exécute le destinataire et quel y est son numéro; il n'y a donc pas de possibilité d'équilibrage dynamique entre plusieurs copies d'un même processus s'exécutant sur plusieurs processeurs, tout est fixé lors de la génération. Les messages sont de deux types: immédiats, ils sont alors rangés et pris en compte selon un algorithme FIFO; avec temporisation, ils ne sont pris en compte que lorsque la temporisation s'annule. Il n'y a pas de priorité.

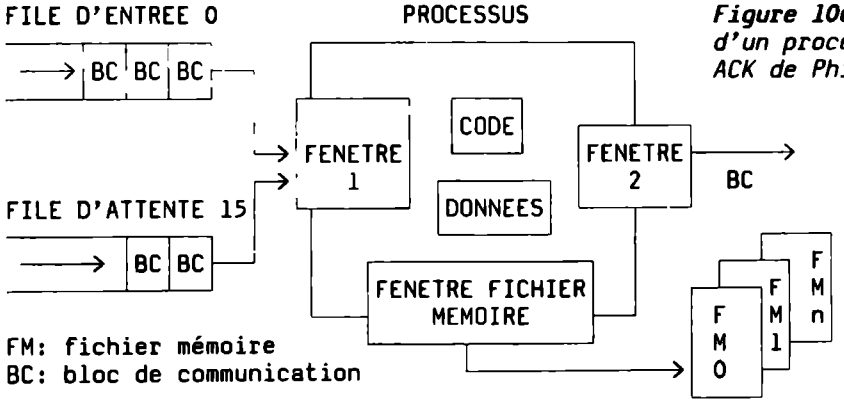


Figure 10u: contexte d'un processus dans ACK de Philips.

Un autre exemple de SE distribué est celui d'ACK (Advanced Communication Kernel) des Engine de Sophonet (Philips) qui réalise les fonctions de:
- distribution du processeur,
- réalisation des E/S physiques,
- gestion du temps (timer),
- gestion des interruptions,

- gestion de la mémoire centrale,
- gestion des communications entre processus,
- journalisation des erreurs et gestion de l'état des processus (bien terminé, fini avec tel type d'erreur, etc.).

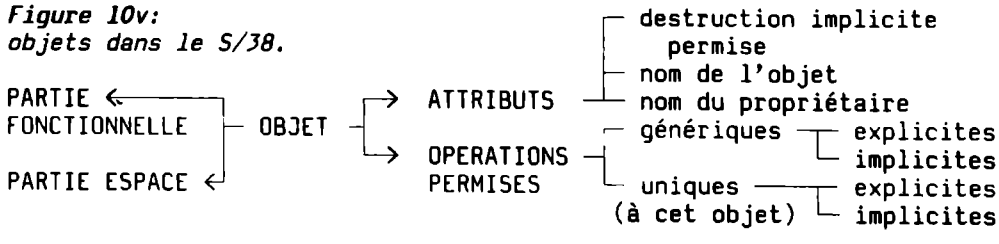
Comme dans Cortex le processus en cours d'exécution n'est pas interruptible au profit d'un autre processus, le processeur n'est pas géré de façon pré-emptive. Les processus communiquent entre eux au moyen de messages ("control blocks" d'au plus 1 Ko), ils se connaissent par leurs **noms** et n'ont pas besoin de savoir sur quel processeur se trouve le processus destinataire, la localisation étant réalisée par le SE. Les contraintes et limitations actuelles de la génération fixent cependant les processeurs dans lesquels les processus seront chargés, sans remettre en cause l'appel par nom. Contrairement à Coralis 80, il n'y a pas de mémoire commune et lorsque deux processus ne sont pas dans le même processeur, il doit y avoir copie des messages et non pas simple modification de pointeurs. Aussi bien dans Cortex que dans ACK il y a duplication du code s'il doit être exécuté dans différents processeurs.

10.11 – Système d'exploitation orienté objet.

Toute l'information du système est structurée en objets classés selon un certain nombre de types. Toute opération sur un objet est contrôlée comme l'est une opération sur un fichier ou sur un sémaphore dans un système classique. Il s'agit d'une réalisation de la notion de type abstrait où un type est un ensemble de valeurs d'objets et d'opérations sur des objets qui en modifient la valeur. Il n'y a alors plus de tables ni de répertoires partagés entre processus, mais des objets partagés dont la structure physique n'est plus connue des processus utilisateurs. Ces derniers ne connaissent que des interfaces qui sont les opérations d'accès [KAI86].

Il y a une centralisation, une unicité des fonctions qui assure une cohérence accrue. On trouve ici la tendance des récents langages "**typés**" comme Pascal, Ada, des processeurs CISC présentés dans le tome I dont l'IBM S/38 est un exemple: le profil de l'utilisateur (ses capacités) y est lui-même un objet; un objet (cf. [PIN78]) est composé d'une partie fonctionnelle (des microprogrammes) et d'une partie espace, il possède des attributs et un certain nombre d'opérations permises.

Figure 10v:
objets dans le S/38.



Les objets du S/38 définis au niveau du matériel sont:
- les groupes d'accès: regroupement d'objets pour faciliter les mouve-

ments entre mémoire centrale et mémoire secondaire;
- les contextes: un répertoire qui permet d'accéder aux autres objets. On y trouve leur type, leur sous-type qui n'est significatif que pour le SE CPF, pas pour le matériel, leur nom;
- les descriptions de contrôleur: des groupes de périphériques;
- les curseurs qui sont utilisés pour adresser l'intérieur des espaces d'adresses;
- les espaces de données qui contiennent un format d'enregistrement de base de données;
- les index d'espaces de données qui définissent l'ordre logique des enregistrements dans un espace de données;
- les index pour stocker et ordonner les données;
- les descripteurs d'unité logique: ils représentent les appareils physiques;
- les descripteurs de réseau: les accès réseau du système;
- les espaces de contrôle des processus: ils contiennent la forme exécutable des processus;
- les programmes;
- les files d'attente (queue) pour la communication entre processus, entre processus et périphériques;
- les espaces pour stocker pointeurs et scalaires;
- les profils des utilisateurs qui identifient les utilisateurs valides du système.

CPF étend ce typage et présente 19 objets ([PIN78]) à l'interface système: les objets de la machine et des objets composites comme les fichiers.

Dans iRMX86 ces types d'objet, tous créés par le SE, sont des:
- processus (task), les objets "sujets" puisque les seuls actifs;
- travaux (jobs), environnements dans lesquels les processus se déroulent. Un environnement comprend des processus, les objets utilisés par ces processus, un répertoire où les processus peuvent cataloguer des objets, un pool mémoire centrale attribué au travail et à ses descendants. La mémoire est découpée de façon statique entre les travaux, à l'intérieur d'un travail elle est répartie dynamiquement entre les processus;
- boîtes aux lettres (mailbox) à travers lesquelles les processus envoient et reçoivent des objets par l'intermédiaire de jetons (token).
- sémaphores pour envoyer des "signaux" aux autres processus;
- segments considérés comme des morceaux de MC, conformément à la philosophie du 8086;
- régions: collections de données partagées;
- objets étendus définis par l'utilisateur;
- utilisateurs dont les processus sont les agents;
- travaux (job) d'E/S;
- appareils logiques;
- connexions.

De tels systèmes sont plus faciles à comprendre, à expliquer ([INT83]), les **attributs** des objets et les **opérations** permises sont clairement définies et cohérentes. Le contrôle de cette cohérence

représente un coût processeur supplémentaire non directement productif qu'iRMX permet de désactiver pour améliorer les performances et diminuer la taille du code.

chapitre 11

gestion de la mémoire centrale

Nous avons déjà introduit la gestion de la mémoire centrale au début de ce tome, pour la définir globalement et la situer dans l'ensemble des ressources que gère le système d'exploitation. Dans ce chapitre nous détaillerons la façon dont le SE gère la mémoire et présenterons la majorité des méthodes utilisées pour gérer cette ressource essentielle. Nous nous plaçons bien entendu dans le cadre de la multiprogrammation, plusieurs processus se partageant la mémoire.

Lorsque l'on parle de mémoire centrale, il convient de distinguer deux notions que nous avons présentées dans le tome I:

- la mémoire **adressable**, elle est définie par l'architecture externe du processeur et correspond au nombre de cases mémoire qu'une instruction peut adresser;
- la mémoire **configurée** qui est la taille de la mémoire centrale installée sur un ordinateur donné, taille dont le maximum est la taille **configurable** qui, elle, est une caractéristique de l'architecture interne du processeur.

MVS a un espace adressable de 16 Mo, la version SP1.3 supporte une mémoire configurable de 64 Mo. VSE/SP2.1 gère une mémoire centrale de 16 Mo, une mémoire virtuelle de 40 Mo. Des contraintes apportées par le SE peuvent limiter ces tailles: ainsi le 8086 a une mémoire configurable de 1 Mo, mais MS-DOS ne permet d'en utiliser que 640 Ko;

11.1 – Partitions fixes (statiques).

Une des méthodes de gestion les plus simples consiste à découper la mémoire en zones contiguës, appelées partitions, dans lesquelles sont rangés chacun des travaux en cours d'exécution. Par partitions fixes nous entendons:

- un nombre fixe de partitions entre deux démarrages (boot, IPL, cf. chapitre 16) ou même deux générations différentes du système ("sysgen" pour "system generation");
- une adresse fixe pour le début de chaque partition;
- et donc une longueur figée pour chaque partition.

Les **avantages** de cette technique résident dans sa **simplicité**: l'allocataire de mémoire exécute un algorithme réduit et peu coûteux.

Les **défauts** en sont:
- une mauvaise utilisation de la mémoire centrale quand certaines partitions ne sont pas utilisées ou du moins pas totalement. Un programme chargé dans une partition n'occupe pas nécessairement toutes les cases de cette partition;
- les travaux acceptables sont limités par la taille de la plus grande partition;
- dans certaines machines une forme exécutable (cas de OS/MFT, DOS d'IBM, voir tome V) ne peut être chargée que dans la partition prévue pour elle lors de l'édition de liens ou même de l'assemblage. Dans les systèmes DOS on en arrivait à préparer plusieurs formes exécutables d'un même programme afin de pouvoir le charger dans la première partition disponible;
- la forme exécutable du programme (PE) doit être totalement chargée en mémoire centrale;
- la mémoire centrale contient donc des zones qui ne sont jamais référencées, parce que tous les cas prévus dans les applications ne se produisent pas nécessairement lors de chaque exécution. C'est le cas des séquences activées en cas d'erreur en particulier;
- cette forme exécutable doit être chargée de façon contiguë en MC.

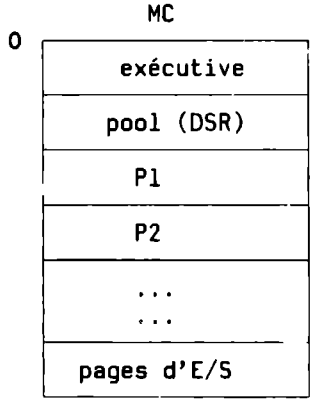


Figure 11a: carte de la mémoire de RSX-11M.

16 Km <= N <= 20 Km (espace du noyau)
DSR = Dynamic Storage Region dont le rôle sera expliqué dans la section sur les pools centralisés.

espace utilisateur découpé en partitions fixes avec possibilité de sous-partitions dynamiques à l'intérieur de la partition

4 Km La DSR contient un certain nombre de blocs de contrôle comme les TCB.

Le terme de partition est utilisé sur DOS et OS/MFT d'IBM, OS/MVT utilisait le terme de "région"; une partie des 52 partitions y sont dites "système" car elles sont utilisées pour des programmes du système comme le GPL (system reader, system writer); seules 15 de ces partitions sont allouables aux utilisateurs du fait des 16 valeurs possibles pour les clés de protection présentées plus en détail dans le chapitre 14. DOS, DOS/VS offraient 5 partitions, en mémoire virtuelle pour DOS/VS; DOS/VSE a 12 partitions.

Sur RSX-11M de DEC, système basé aussi sur des partitions fixes, les processus dits avec "points de contrôle" (checkpointing tasks) peuvent cependant être temporairement vidés de MC sur disque pour laisser la place à un processus plus prioritaire. Le processus sera ensuite rechargé en MC lorsque sa partition redeviendra libre.

11.2 – Partitions de longueur variable.

Dans les systèmes utilisant des partitions variables comme technique de gestion mémoire, la contrainte sur la taille maximale de la partition est supprimée, ce qui permet à de gros travaux de pouvoir être logés en MC. RTE-A utilise ainsi un système de partitions variables qui permet à une partition de s'étendre dans la partition suivante.

La contrainte sur l'adresse d'implantation de la partition peut aussi être levée. Ainsi dans OS/MVT d'IBM les "régions" pouvaient commencer à une adresse quelconque; dès le début de l'exécution la région devait avoir la taille maximum atteinte par le travail au cours de l'exécution (paramètre REGION du JCL); son adresse de chargement ne pouvait plus changer jusqu'au terme du travail; par la suite ce fut la taille de chaque étape qui intervint. Une option de vidage sur mémoire secondaire (disque) apparut aussi sous MVT (roll-in/roll-out) mais se révéla d'emploi délicat puisqu'il fallait obligatoirement recharger à la même adresse un programme vidé. Le CO500 de Sfena sous le SE Meteor utilise aussi un système de partitions non translatables une fois le processus chargé.

11.3 – Partitions translatables (dynamiques).

Le terme de translatable a deux incidences dans ce chapitre, et on parlera de:

- translatabilité de chargement lorsque la forme exécutable peut être chargée n'importe où en mémoire centrale;
- translatabilité d'exécution si l'on peut déplacer la forme exécutable en mémoire centrale après qu'elle ait commencé à exécuter ses instructions;

Les partitions sont en nombre variable, démarrent n'importe où en MC et sont de longueur variable. Il faut pour cela que le(s) registre(s) de base contenant les adresses de début de partition ne soient pas accessibles en mode utilisateur: s'ils le sont sur les IBM/360, ils ne le sont pas sur les systèmes comme GCOS3, OS/1100, CYBER, DEC10, ICL1900, B4800.

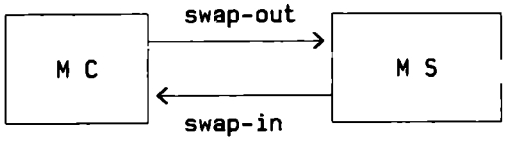


Figure 11b: cadre du va-et-vient (swapping).

MC: mémoire centrale
MS: mémoire secondaire (disque)

Les champs adresse logique AL des instructions contiennent des déplacements relatifs par rapport au début de la partition, d'où la translatabilité. Le SE doit charger le registre de base avec l'adresse de la partition à chaque commutation de processus. Le 8086 en offre un bon exemple, mode privilégié et commutation de processus mis à part.

Ce type de gestion MC conduit à créer des trous en MC en dehors des partitions: on parle alors de fragmentation externe. Pour faire

place à un nouvel arrivant il faut lui trouver une place, celle qui convient le mieux (best fit, optimum du point de vue de la fragmentation externe), la première qui convient (first fit, optimum sur le plan de la vitesse), celle qui conduit au plus grand fragment (worst fit), selon les algorithmes utilisés (cf. [BOZ84]).

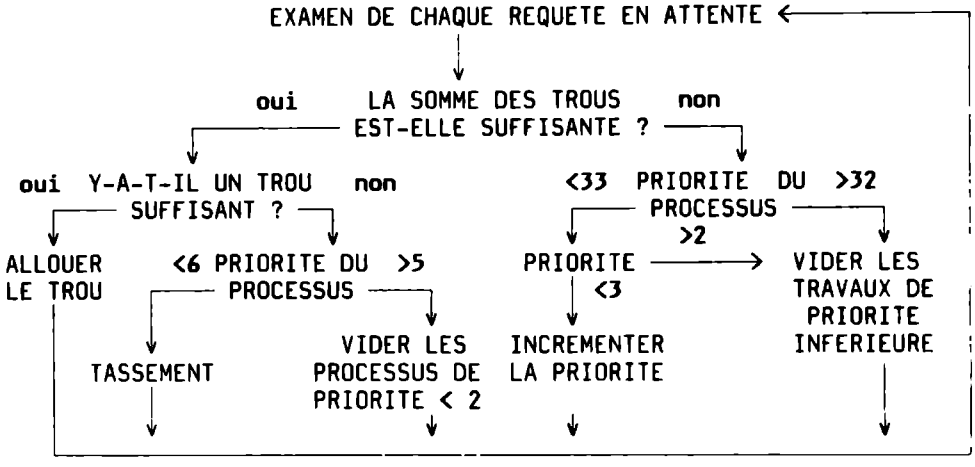


Figure 11c: algorithme de gestion MC utilisé par une des versions de GCOS3; il est applicable à GIN et PALC dont la priorité est égale à zéro en inactivité, à 62 sinon (le maximum est de 63).

Si aucun trou ne convient, le gestionnaire MC opérera des **tassements** en MC afin de créer un trou suffisamment grand pour loger le nouveau processus (cf. figure 11c). Si les tassements ne suffisent pas, il faudra en arriver au vidage sur disque des processus moins prioritaires: méthode de **va-et-vient** (swapping, roll-out/roll-back, roll-in), mise en oeuvre comme on l'a vu plus haut dans certains systèmes à partitions fixes comme RSX-11M de DEC. Les mémoires centrales gérées par segmentation présentent les mêmes caractéristiques.

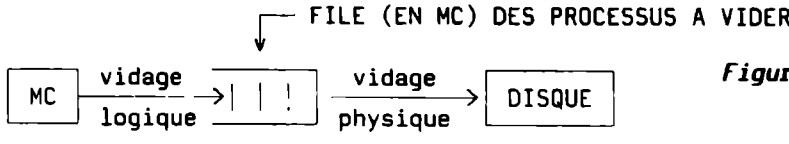


Figure 11d.

On parlera aussi de "**vidage logique**" (logical swap) lorsque le processus est mis dans la file d'attente des processus à vider sur mémoire secondaire, mais sans que le vidage soit réellement effectué: le processus reste en MC. Quand il faut le ramener en MC, s'il se trouve encore dans la file, on l'enlèvera simplement de cette file. Le "vidage physique" (physical swap) correspond, lui, à une écriture réelle sur disque puis à une lecture ultérieure du disque lorsque les

conditions permettent le rechargement (swap-in) du processus (figure 11b).

Nous verrons avec le sous-système temps partagé de MVS (TSO) un cas de vidage logique. Dans VMS le vidage est géré par le processus appelé "swapper"; il dispose d'ailleurs de la priorité 16, frontière entre les processus interactifs et les processus temps réel. Dans Unix, le vidage peut être réalisé par le noyau.

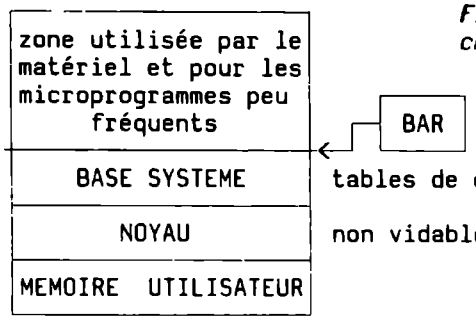


Figure 11e: carte mémoire d'un DPS7 où chaque segment constitue une partition translatable pour la gestion de la mémoire.

BAR: Base Address Register

11.3.1 – Partitions avec recouvrements système.

Dans le chapitre sur l'évolution nous avons présenté le superviseur découpé en deux parties en mémoire centrale:
- une partie résidente,
- une partie réservée aux recouvrements du système.

Cette deuxième zone de recouvrements devient difficile à gérer dans un environnement avec un taux de multiprogrammation élevé: les différents processus ont des besoins différents qui se télescopent et dépassent les possibilités de la zone de recouvrement du système. Pour résoudre ce problème on peut:
- augmenter le nombre de zones de recouvrement du système,
- distribuer ces zones de façon à ce que chaque processus porte avec lui les modules système dont il a besoin à un instant donné.

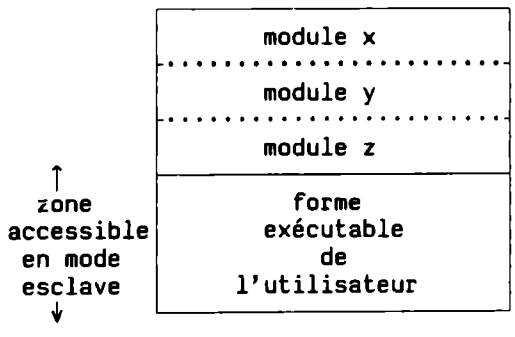


Figure 11f: structure d'une partition dans GCOS3.

C'est le second choix qui a été fait dans GCOS3, la structure de la machine faisant que les modules système chargés dans la SSA ne peu-

vent pas être écrasés par les instructions de l'utilisateur (cf. figure 11f).

Dans une des versions de GCOS3 a été introduite la DCA (Disk Cache Area) dans les adresses hautes de la mémoire afin de diminuer les E/S provoquées par le chargement des modules système vers les zones SSA: les modules système les plus sollicités sont conservés dans cette zone DCA, gérée comme une pile. Ils sont copiés dans les zones SSA lorsqu'ils sont appelés, et non plus chargés depuis les disques. Cette pile se comporte comme une sorte d'antémémoire vis-à-vis du disque système.

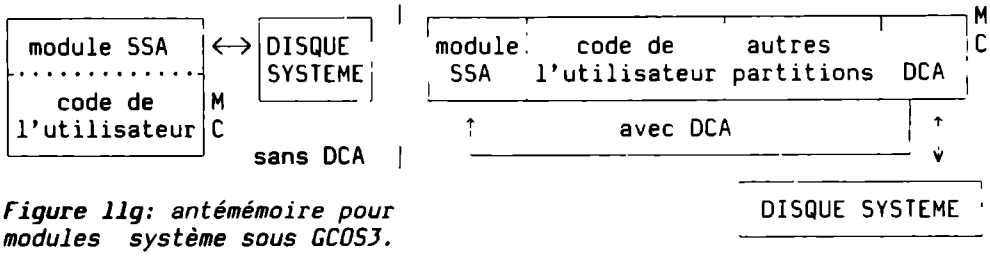


Figure 11g: antémémoire pour modules système sous GCOS3.

Lorsqu'il faut vider (swap) un processus sur disque, le SE n'a pas à se demander si on peut en profiter pour enlever de ses zones de recouvrements les modules chargés pour ce processus: il vide automatiquement et la partie utilisateur et la partie système utilisée.

On retrouve une approche analogue avec la LPA (Link Pack Area) de OS/MVT ou avec la SVA dans la mémoire virtuelle de DOS/VS. La LPA de MVT pouvait ainsi contenir les modules de gestion de la mémoire, du planificateur (initiator/terminator), des routines réentrantes de l'utilisateur, des répertoires pour les librairies du système, le répertoire des routines SVC, des méthodes d'accès.

11.3.2 – Partitions avec code et données séparés.

Une forme exécutable se retrouve chargée en mémoire dans deux partitions indépendantes, l'une pour les instructions (partition Ii), l'autre pour les données (partition Di). Chaque partition est protégée par un double jeu de registres de base et limite:

[RBI,RLI] , [RBD,RLD]

Une même partition de code (I1 dans l'exemple dans la figure 11h) peut travailler sur des jeux de données différents, sans avoir à être dupliquée en mémoire:

- le travail T1 aura comme code I1 et comme données D1,
- le travail T3 aura comme code I1 et comme données D3.

Si plusieurs utilisateurs lancent une compilation Cobol simultanément, le code du compilateur ne sera chargé qu'une seule fois en mémoire. La séparation du code et des données est une nécessité pour assurer la réentrance (cf. tome V).

L'OS/1100, les PDP10, disposent de cette séparation entre code et données. Une des MMU (Memory Management Unit) des PDP-11 permet de séparer l'espace des instructions et l'espace des données, chaque par-

tition peut alors occuper 64 Ko, et l'ensemble atteint ainsi 128 Ko au lieu des 64 Ko normalement adressables sur les PDP-11.

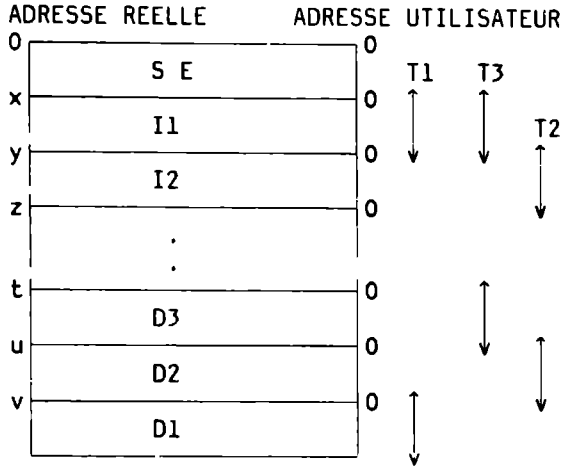


Figure 11h: carte mémoire avec partitions séparées pour le code et les données.

processus représentés:
. T1: [I1,D1]
. T3: [I1,D3]
. T2: [I2,D2]

11.3.3 – Partitions multiples.

Afin de faciliter le chargement d'un travail en mémoire, en surmontant le problème des trous, on lui alloue plusieurs partitions. Cette solution est parfois appelée, à tort, segmentation, et constitue une généralisation des partitions avec code et données séparées vues ci-dessus.

On trouve ce type de solution sur les mini-ordinateurs DS990 de Texas Instruments dans lesquels une forme exécutable peut être éclatée en MC en quatre partitions, ceci étant géré directement par le matériel. Dans ce cas le contexte du processus se voit augmenté d'un certain nombre de registres donnant l'adresse du début de chaque partition. IBM avait essayé ce type de solution sur 360 avec l'OS/VMS (partitions multiples statiques) qui ne fut jamais commercialisé, OS/MVT ayant été jugé plus performant.

11.3.4 – Remarques sur les recouvrements.

Si l'on veut exécuter un processus dont la taille dépasse celle de la partition dans laquelle il a le droit d'être chargé, il faut le découper en morceaux:

- une racine qui résidera dans la partition pendant toute l'exécution du processus,
- des branches qui seront appelées par la racine lorsqu'elles seront nécessaires.

L'utilisateur est ainsi obligé de gérer lui-même la mémoire.

Les différentes branches (Bi sur le schéma 11i) seront chargées dans la même zone MC (elles se "recouvriront", overlay) depuis le disque sur lequel elles résident (cas de TDS sous GCOS3 par exemple).

Dans certains systèmes, les branches pourront résider en MC et

être donc amenées plus rapidement dans les partitions où se trouve la racine (certaines versions japonaises de TDS sous GCOS3 par exemple): le temps de chargement, une copie en MC en fait, est beaucoup plus rapide.

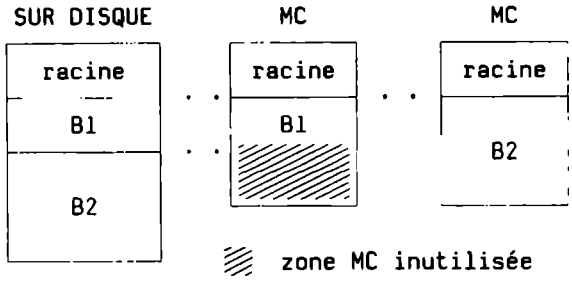


Figure 11i: recouvrements classiques gérés par l'utilisateur dans sa partition.

Pour économiser le temps processeur représenté par le déplacement de ces branches en MC, des milliers d'octets, certains systèmes disposent de registres de traduction d'adresse (voir plus loin): il suffit de modifier le contenu de ces registres pour changer la branche réellement adressée; sur un 8086 on modifierait le contenu des registres de base des segments. On trouve cette possibilité sous RSX-11M de DEC par exemple: on y parle d'ailleurs de mémoire virtuelle !

11.4 – Pools centralisés.

La gestion par pools centralisés est fréquemment utilisée dans les systèmes temps réel ou même interactifs, là où la charge est variable et où il est difficile de prévoir ce que chaque processus va consommer. Il y a deux grands types d'algorithmes: ceux qui allouent exactement la taille demandée, conduisant ainsi à une fragmentation qui est source de nombreux problèmes, et ceux qui découpent la mémoire en blocs de taille identique.

Si les tailles des requêtes sont très variables, certains algorithmes n'utiliseront qu'une seule taille de bloc et alloueront plusieurs blocs chaînés lorsque nécessaire, d'autres géreront plusieurs pools avec des blocs de tailles différentes pour chaque sous-pool (buddy system de [BOZ84]), et chaque demande sera satisfaite avec le plus petit bloc de taille égale ou supérieure à celle de la demande.

Les algorithmes plus sophistiqués sont capables d'adapter dynamiquement la taille de chaque sous-pool à la charge instantanée du système, découpant des gros blocs en petits blocs, regroupant des petits blocs en gros blocs si nécessaire.

En cas de saturation, c'est-à-dire lorsque le nombre de blocs disponible tombe en-dessous d'un seuil prédéfini, il y a soit blocage du système, soit ralentissement (slowdown) s'il est suffisamment évolué. Ainsi dans NCP, le logiciel des frontaux 37X5 d'IBM, deux types de requêtes, conditionnelles et inconditionnelles, permettent de préciser si l'utilisateur maintient sa requête en cas de saturation ([NCP83]).

Cette technique des pools centralisés peut être combinée avec les

autres techniques de gestion de la mémoire comme le montre le cas de RSX-11M figure 11a. Le système des partitions satisfait le besoin minimum en mémoire, les suppléments sont acquis de façon dynamique. Une bonne analogie avec la gestion des pools centralisés est offerte par la gestion de la consigne dans une gare.

11.5 – Pagination.

La pagination supprime la contrainte de contiguïté de la forme exécutable en MC. La MC est découpée en blocs de longueur fixe appelés **cadres** qui contiennent la forme exécutable découpée en **pages** de même longueur. Ce mécanisme est **transparent** pour le programmeur: il est géré par le matériel et le système d'exploitation, le programmeur ne voit à aucun moment les pages de sa forme exécutable; ce découpage est totalement arbitraire en ce qui le concerne et une instruction pourra avoir ses différents octets répartis sur deux pages consécutives.

Nous utiliserons la terminologie suivante, analogue à celle de MVS:

- **cadre** (frame) pour un bloc en MC,
- **case** (slot) pour le bloc correspondant sur disque,
- **page** (page) correspondante dans la forme exécutable PE. La page sera stockée dans une case et éventuellement dans un cadre comme nous l'exposerons plus loin.

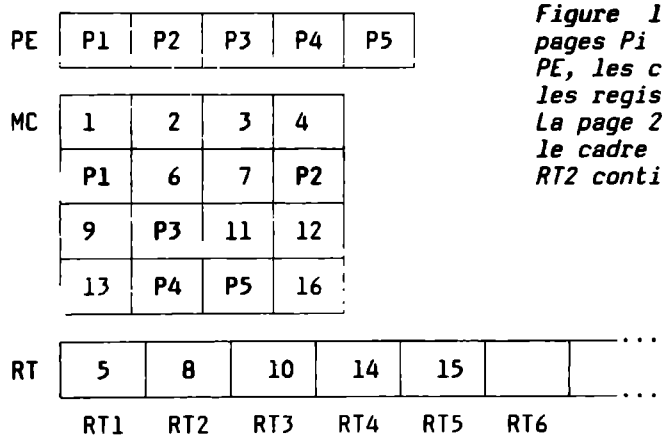


Figure 11j: relations entre les pages Pi d'une forme exécutable PE, les cadres de la mémoire et les registres de traduction RTj. La page 2 (P2) est chargée dans le cadre 8, ce qui explique que RT2 contienne 8.

11.5.1 – Support physique.

La forme exécutable n'étant plus totalement contiguë en mémoire centrale, il faut disposer de circuits permettant de traduire (translate) l'adresse "linéaire" vue par le programmeur en adresse physique correspondant au rangement réel du programme en MC.

Cette traduction (DAT, Dynamic Address Translation chez IBM) est réalisée par des circuits spécialisés comportant notamment des **registres de traduction** (TLB pour translation registers, TransLation Buffer chez IBM, TB, DLAT, etc.) que nous abrégerons en RT dans la suite. Ces

circuits spécialisés contenant les RT sont génériquement appelés MMU (Memory Management Unit, unité de gestion mémoire) dans les microprocesseurs.

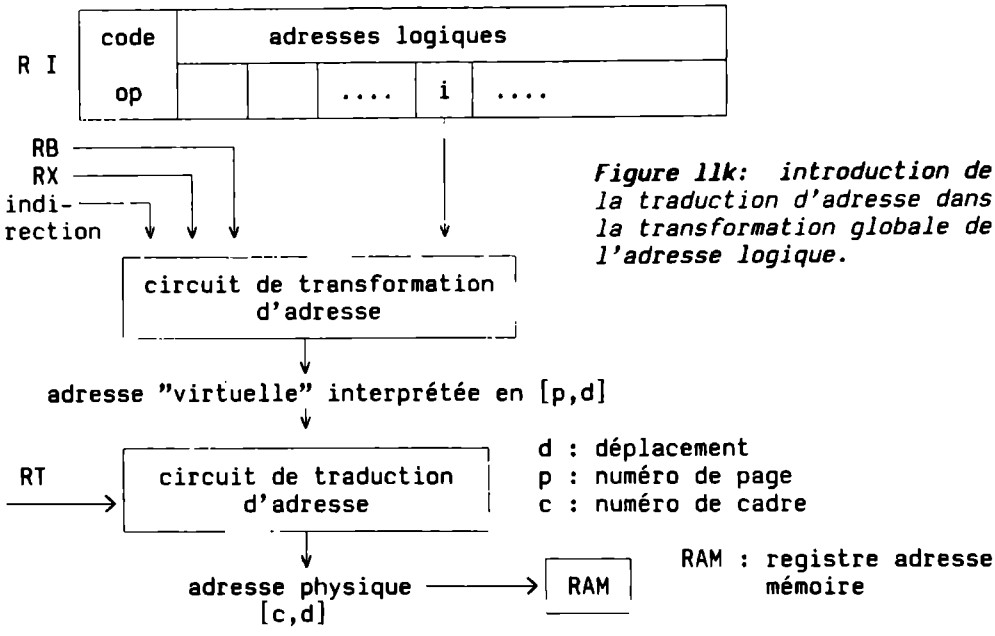


Figure 11k: introduction de la traduction d'adresse dans la transformation globale de l'adresse logique.

Les circuits de traduction d'adresse peuvent être désactivés: dans le cas du 370 d'IBM par exemple, si le bit T du mot d'état de la machine (PSW, Program Status Word) est égal à zéro lorsqu'elle est en mode étendu (C=1 dans ce même PSW, le mode étendu signifie que la machine travaille en 370 et non en 360) il n'y a pas de traduction d'adresse.

Sur certains ordinateurs, l'ensemble des registres de traduction est divisé en sous-ensembles spécialisés:

- la moitié des RT du **VAX-11/780** est réservée aux processus des utilisateurs (les applications), l'autre moitié est réservée pour le système. Ainsi seuls ceux réservés aux processus des utilisateurs sont effacés (invalidés, flush, cf. [CLA85]) quand il y a commutation de processus. Les mesures de [CLA85] montrent que ce partage est efficace lorsque le nombre de RT est suffisamment grand. Les instructions TBIS et TBIA invalident un RT ou tous les RT. Dans le cas de la pagination dynamique que nous introduirons ci-après, il faut effacer l'entrée correspondant à la page "volée" (rôle de l'instruction IPTE dans les 370) dans les registres de traduction de chaque processeur, sinon l'ancien processus qui la "possédait" accéderait à une page d'un autre processus;
- dans le **Z8108** il est possible de spécialiser une partie des RT pour les instructions, l'autre pour les données. Ils sont par ailleurs

associés soit au mode système, soit au mode utilisateur, de façon analogue au VAX.

S'il est assez répandu, le vidage total des RT lors de la commutation de processus n'est pas universel: dans le 90X de Pyramid chaque RT contient l'identification (hash) du processus auquel il est associé ([BOT84]).

Dans le VAX les RT sont organisés en n niveaux, analogues aux niveaux que nous avons décrits pour les antémémoires. Quand un RT est requis, le choix ne se fait pas sur l'ensemble des RT, mais sur n RT seulement, les seuls susceptibles de satisfaire la demande. La colonne A comme associativité du tableau 11k donne quelques exemples de valeur de n.

SYSTEME	NOMBRE DE RT			
	TOTAL	A	SYSTEME	UTILISATEUR
VAX 8600	512	1	256	256
VAX-11/785	512	2	256	256
VAX-11/780	128	2	64	64
VAX-11/750	512	2	256	256
VAX-11/730	128	1	64	64
Z8108	32		16	16
IBM 3031	128		banalisés	
64 (2)	8		banalisés	
370/138	8		banalisés	
DPS7 (2)	128		banalisés	
NS 16000 (1)	32		banalisés	
GE645	16		banalisés	
360/67	8		banalisés	
Z80000 (3)	16		banalisés	
iAPX386 (3)	32		banalisés	
DPS8	64		banalisés	

Tableau 111: exemples de registres de traduction.

A: associativité

Les RT système sont utilisés par les processus du système.

(1) dans la MMU 16082
(2) segmentation en fait
(3) sur la puce (on chip) MMU 68451 avec le 68010

En résumé, l'adresse logique subit ainsi deux transformations successives et de nature différente:

- transformation liée aux **modes d'adressage** vus dans le tome I: elle a des objectifs orientés vers la réduction de la taille des programmes, vers le maximum de cases adressables, vers la facilité de programmation, etc. Cette transformation est visible du programmeur;
- transformation liée à la **gestion de la mémoire** centrale. Cette seconde transformation est invisible (transparente) pour le programmeur.

Notons que l'ajout de cette traduction correspond à un coût accru du fait du matériel supplémentaire, ainsi qu'à une perte de temps due à la traduction. Selon [CLA85] le VAX-11/780 consacre entre 5 à 8 % de sa puissance processeur pour gérer ses RT; sur le plan des coûts, ils occupent 1,3 des 20 cartes du processeur qui a une technologie datant

des années 1975-1976, soit 6 à 7 % du coût du processeur.

Dans le cas des architectures pipelinées (voir tome I), le calcul des adresses virtuelles des opérandes peut se faire avant le début de l'exécution des instructions (MV/10000 de Data General par exemple). [LUK86] présente une étude sur les RT du processeur RISC qui sert de base aux nouveaux modèles HP3000, HP9000.

11.5.2 – Cas des canaux.

Dans la plupart des machines, les canaux (ou les processeurs d'entrée/sortie, cf. tome III) n'ont pas accès aux dispositifs de traduction d'adresse, ni aux RT. Avant de lancer une E/S, le SE doit donc traduire les adresses contenues dans le programme canal en adresses physiques et l'on parlera d'absolutisation des adresses, rôle de l'instruction ACCE du DPS7 par exemple.

Dans les 308X, les 3090, dans les DPS8 en mode virtuel, les organes d'E/S ont accès aux RT et/ou aux tables de pages et de segments que nous verrons plus loin.

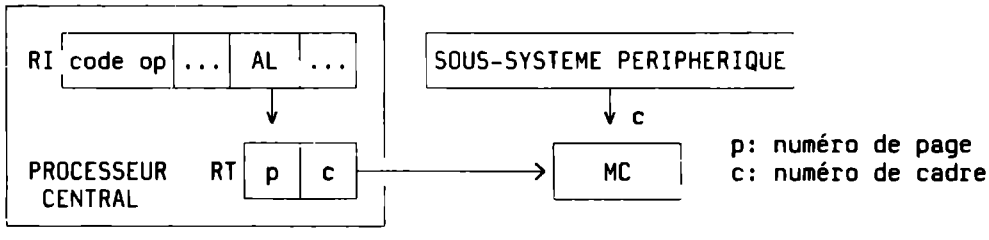


Figure 11m: les sous-systèmes périphériques n'ont pas toujours accès aux adresses virtuelles.

11.5.3 – Lien avec l'antémémoire.

Nous avons vu dans le tome I le rôle joué par l'antémémoire qui utilise une partie de l'adresse physique pour chercher dans ses mémoires internes si elle dispose de l'information recherchée, évitant ainsi à la requête d'aller jusqu'à la mémoire centrale pour être satisfaite, d'où une plus grande rapidité globale au niveau du sous-système central. L'antémémoire, abrégée en AM par la suite, est dite associative car sa recherche est basée sur une partie de son contenu: la partie de l'adresse physique sur laquelle se fait la recherche.

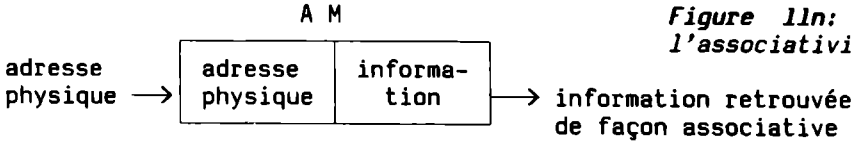


Figure 11n: rappel de l'associativité des AM.

Nous avons donc le cheminement général suivant illustré par la figure 11o.

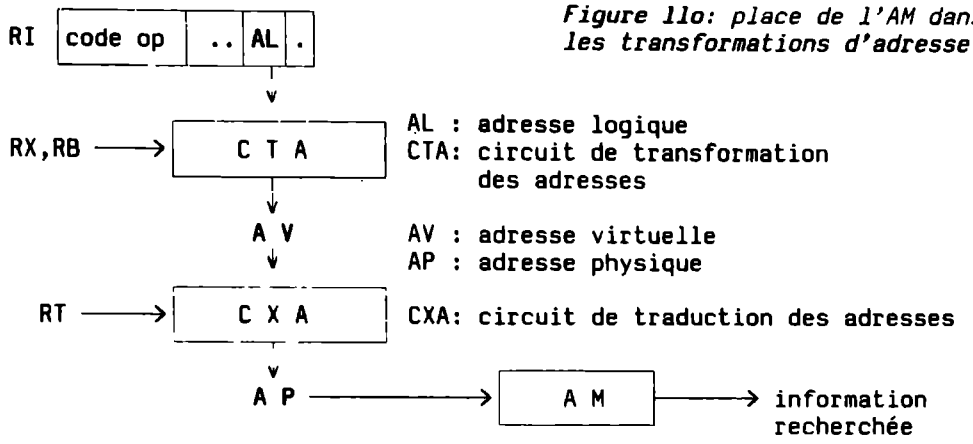


Figure 11o: place de l'AM dans les transformations d'adresse.

Dans certains ordinateurs l'AM reçoit en entrée non pas l'adresse physique, mais une adresse virtuelle (figure 11p). C'est le cas de l'ICL 2900, des IBM 3090 ([TUC86A]), de certaines machines expérimentales comme le MU-5, le S-1, l'IBM 801, etc. Ce mécanisme est plus rapide puisque la traduction n'est pas requise, mais il peut provoquer quelques problèmes, nécessitant parfois la traduction inverse (de virtuel en réel), traduction réalisée par des RTB (Reverse Translation Buffer). Il faut aussi résoudre le problème des synonymes, c'est-à-dire des différentes AV qui correspondent à la même AP.

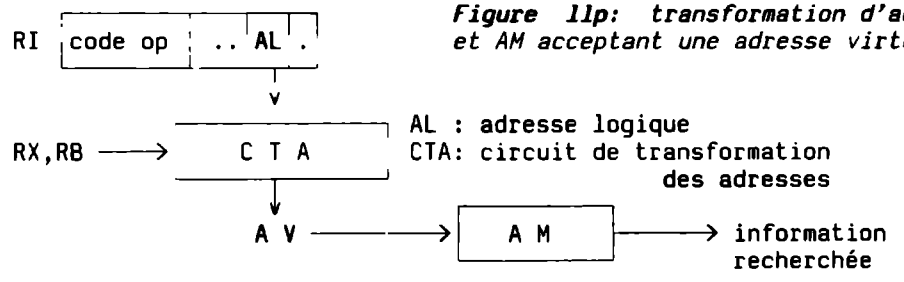


Figure 11p: transformation d'adresse et AM acceptant une adresse virtuelle.

11.6 – Les types de registres de traduction.

On peut introduire la pagination comme une évolution ultime des partitions multiples. Nous distinguerons deux grandes techniques:

- la **carte mémoire complète** (map, mémoire topographique): il y a autant de registres de "traduction" que de cadres dans la mémoire centrale configurée. C'était le cas de la 10070 de CII (version française du Sigma 7 de SDS puis XDS) qui disposait de 256 registres de 9 bits permettant d'adresser 128 Km. Cette technique est généralement associée à une pagination statique;
- **la carte mémoire restreinte:**
  * à l'espace adressable par un processus: on trouve souvent cette

solution sur des minis de 16 bits où la mémoire maximale initiale était de 32 ou 64 Ko; ces minis ont vu leur MC augmenter (jusqu'à plusieurs Mo) pour faciliter la multiprogrammation; l'espace directement adressable par un processus (32 ou 64 Ko) n'est pas accru pour autant. Les registres de traduction font alors partie du contexte du processus et doivent donc être sauvegardés-restaurés à chaque commutation de processus. C'est le cas du PDP11 (le PDP11/70 a ainsi 8 RTs appelés APR, pour Active Page Registers, et 16 si l'on utilise la séparation du code et des données), de l'Eclipse de Data General; l'Interdata 7/32 en dispose de 16.

64 RT/code système	64 RT/données système
64 RT/code utilis.	64 RT/données utilis.

Cas du Tandem16.

← invalidés à chaque commutation de processus

Tandem16 (machine 16 bits avec adresses de 16 bits) est dans un cas analogue, il dispose de quatre jeux de RT: les deux premiers sont réservés au SE, un pour le code, l'autre pour les données puisqu'il y a séparation du code et des données, les deux autres pour le processus actif, l'un pour le code et l'autre pour les données; chaque jeu comporte 64 RT ce qui permet d'adresser 64 pages de 2 Ko, soit 128 Ko; un processus a donc un espace adressable de 256 Ko découpé en deux parties égales pour le code et les données; les deux jeux de RT du processus actif sont bien entendu sauvegardés à chaque commutation de processus;

* par un jeu de registres de "traduction" partagés successivement (cas fréquent) ou simultanément (cas du 90X de Pyramid) par l'ensemble des processus en cours d'exécution. Lorsque la mémoire centrale et l'espace adressable sont importants, il n'est plus réaliste, économiquement, d'équiper l'ordinateur d'autant de registres de traduction qu'il y a de pages ou même de cadres. Tout comme l'antémémoire contient les mots mémoire utilisés durant une courte période, les registres de traduction en nombre restreint contiendront les adresses des pages référencées pendant un laps de temps donné. Les registres de traduction forment ainsi une sorte d'antémémoire vis-à-vis des tables de page: AM et RT sont des "capteurs de localité".

La solution des registres partagés est toujours celle choisie pour la pagination dynamique. Lorsque les instructions du processus en cours d'exécution produisent des adresses, de code ou de données, deux cas peuvent se produire comme le montre la figure 11r:

- un registre de traduction permet de passer de l'adresse virtuelle (p,d) à l'adresse physique (en un cycle de 80 ns sur 370/168),
- aucun registre ne permet de traduire l'adresse (échec, TLB miss) Il faut alors:
  * trouver un RT disponible;
  * recourir à une table des pages TdP (page table) qui se trouve en

mémoire centrale et contient pour chaque page, son adresse physique de chargement en mémoire, telle que le chargeur l'a remplie, lors du chargement de la forme exécutable. La table des pages joue ainsi le rôle de la mémoire topologique vue plus haut;

* copier la p-ème entrée de la table des pages dans le RT choisi, ce qui prend entre 8 et 26 cycles sur un 370/168. Un 3031 nécessitera de 6 à 26 cycles selon l'emplacement des entrées des tables de pages requises pour la traduction. Sur un VAX-11/780 [CLA85] montre que 22 cycles de 200 ns sont nécessaires pour résoudre une faute de RT, temps d'exécution du microprogramme concerné.

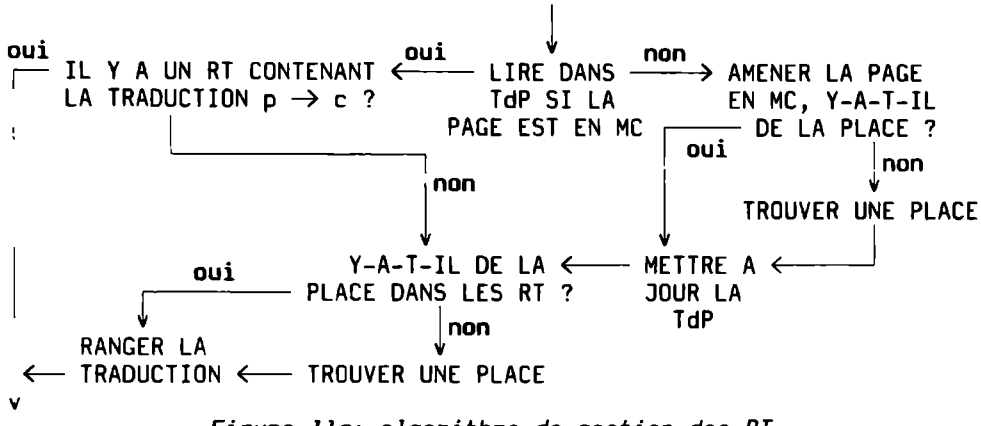


Figure 11q: algorithme de gestion des RT.

A la différence des autres solutions, les registres de traduction partagés forment une mémoire de type **associatif** puisque chaque registre contient et le numéro de la page et son adresse d'implantation en mémoire centrale:

page p	cadre c

Les ordinateurs disposent de quelques RT à quelques centaines de RT (de 8 à 128, selon les modèles, sur la série 370 d'IBM) comme le montre le tableau 111.

Dans le tome I nous avons fonctionnellement partitionné le processeur en trois parties dont l'Unité de Stockage (US) qui contient l'antémémoire AM et les registres de traduction RT. Nous présenterons plus loin les RT comme une antémémoire vis-à-vis des tables de pages ou de segments, leur efficacité se traduisant par un taux de succès (hit ratio) analogue à celui des antémémoires. Les mesures publiées par [CLA85], sur un VAX-11/780 utilisé en développement de programme, donnent des taux de succès de 99 % pour les RT utilisateur, et 94 % pour les RT système, le taux d'échec étant deux fois plus élevé pour les données que pour les instructions, ce qui dénote un meilleur effet de localité pour ces dernières. [CAS78] avance 99 % pour les 370 et Intel annonce 98 % pour l'iAPX386.

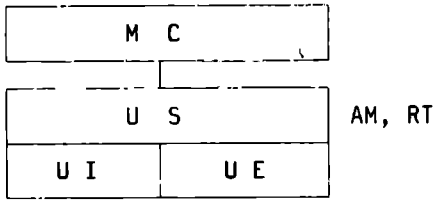


Figure 11r: rappel de la structure du processeur central.

US: unité de stockage
UI: unité d'instruction
UE: unité d'exécution

Nous distinguerons deux grands types de pagination:
- **statique**: la forme exécutable n'est pas contiguë en MC, mais elle y réside totalement. Le chargeur doit toujours la ranger complètement en MC avant qu'elle puisse dérouler ses propres instructions;
- **dynamique**: non seulement la forme exécutable n'est plus contiguë en MC, mais en outre elle n'a plus besoin d'y résider complètement pour dérouler ses instructions. On parlera alors de **mémoire virtuelle** et d'adresse virtuelle puisque tout n'est pas en mémoire centrale, appelée **mémoire réelle** en la circonstance. L'adresse identifie un morceau de code ou de données mais ne décrit plus où est stockée l'information comme précédemment. Elle sera notée:

  p,d

  avec "p" le numéro de la page,
  et "d" le déplacement dans la page.

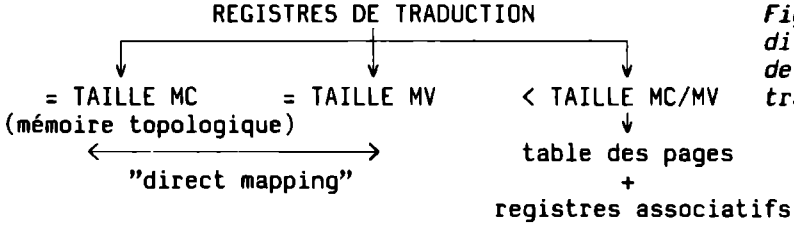


Figure 11s: les différents types de registres de traduction.

11.7 – Pagination statique.

Par rapport aux partitions translatables la pagination statique présente les **avantages** suivants:
- disparition de la fragmentation externe de la mémoire centrale;
- disparition du besoin de tassement qui représente une perte au niveau du processeur central;
- accroissement du taux de multiprogrammation puisque la mémoire est mieux utilisée et peut donc contenir davantage de processus;
- l'élimination des morceaux de mémoire défectueux est facilitée.

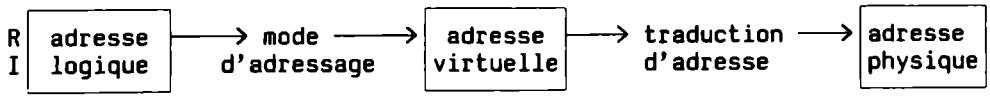
Quant aux **inconvénients**:
- le matériel qui réalise la traduction des adresses se traduit par:
  * un coût supplémentaire,
  * un ralentissement potentiel,
- les tables de page:
  * occupent de la place en mémoire centrale,
  * occupent le processeur central pour leur gestion,

- une fragmentation interne apparaît: certaines pages ne sont pas totalement utilisées:
  * d'où une tendance vers des pages de petite taille,
  * sans atteindre des pages trop réduites car les tables de page voient leur taille augmenter parallèlement,
- si la taille globale des cadres disponibles en MC est inférieure à la taille de la forme exécutable à charger, ces cadres resteront inutilisés;
- la MC contient encore du code non utilisé ou rarement utilisé (cas des séquences traitant les erreurs par exemple);
- une forme exécutable ne peut adresser qu'une zone (on dira un espace d'adresses, un espace adressable, adress space) de taille inférieure à celle de la mémoire centrale configurée.

En dehors des systèmes à mémoire topologique signalés plus haut, le système 36 d'IBM utilise aussi la pagination statique. Sintran III de Norsk Data parle de mode "non demand" pour la pagination statique.

Nous terminerons cette section en donnant quelques exemples de tailles de page:

- 512 octets sur VAX, Classic de Modcomp, NS16032, 3B5 d'AT&T (512 ou 2048), le S/38 d'IBM;
- 1 Ko sur Z80000, Domain d'Apollo, ICL 2900;
- 2 Ko sur 370 avec DOS (la taille est indiquée dans le registre de contrôle CR0), PRIME, PDP10 et PDP20 avec pagination (pages de 512 mots de 36 bits), Xerox Sigma 7, MV/X000 de Data General, HP1000 où la page correspond à une extension de la notion de secteur telle que l'a présentée le tome I dans le chapitre sur les modes d'adressage, 90X de Pyramide, S/36 d'IBM, Tandem16;
- 3 Ko sur les Harris avec Vulcan;
- 512 mots de 36 bits sur DEC PDP-10, PDP-20;
- 4 Ko sur 370 avec les OS d'IBM, sur Z800 sans séparation code-données, sur RCA Spectra 70/46, sur CDC Star-100, GEC63, iAPX386, Z8018, 1 Km de 36 bits sur DPS8 avec GCOS8;
- 64 mots (pour le code) ou 1 Kmot (pour le code ou les données) de 36 bits sur Multics;
- 8 Ko sur PDP11/70, PDP11/45, sur Z800 lorsqu'il est dans le mode séparant code et données. En fait la taille varie de 64 octets à 8 Ko (1 à 128 "blocs" de 64 octets) par incréments de 64 octets sur PDP11. Il en va de même sur les mini 16 bits de Texas Instruments où la taille des pages est variable et correspond plus à une technique de partitions multiples translatables.



11.8 – Pagination dynamique.

Avec la pagination dynamique nous entrons dans le domaine des mémoires virtuelles que nous détaillerons dans la prochaine section. Pour s'exécuter, une instruction n'a plus besoin d'être assurée au

départ de la présence de ses opérandes en MC. Lorsqu'une adresse virtuelle est formée deux cas peuvent se produire:
- les zones référencées se trouvent en MC,
- une au moins des zones référencées n'est pas en MC: on parle alors de **faute de page** (page fault, défaut de page) et il va falloir aller chercher la page concernée (sur disque en général, d'où le nom de mémoire virtuelle, de "backing store", pour le(s) disque(s) contenant les pages) et lui trouver une place en MC. Pour trouver où réside la page on se sert là encore de la table des pages dont le format de l'entrée est donné ci-dessous:

V	adresse

V: bit de validité

V = 1 —> la page correspondante est en MC à l'emplacement désigné par le champ "adresse", l'entrée est "valide",
V = 0 —> la page n'est pas en MC, mais sur disque à l'emplacement indiqué par le champ "adresse", l'entrée est "invalide".

Une instruction peut ainsi être interrompue à cause de l'absence de l'un de ses opérandes puis être relancée (restart) une fois l'opérande amené en mémoire. Tous les processeurs ne supportent pas la relance transparente d'une instruction ainsi interrompue: les 68010 et 68020 oui, le 68000 non.

Le **taux de pagination** (page fault rate) exprime le nombre de fautes de page par seconde. On distingue généralement:
- la **pagination d'entrée** (page-in) qui a trait aux pages chargées en mémoire centrale depuis le disque, donc aux fautes de page;

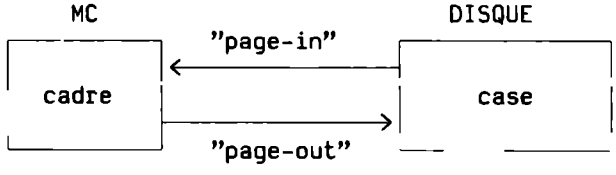


Figure 11t: pagination en entrée et en sortie.

- la **pagination de sortie** (page-out) qui correspond aux pages vidées de MC sur disque: il faut en effet faire de la place en MC pour les pages demandées (d'où le nom de pagination à la demande, demand paging) par les instructions qui se déroulent, mais ce chargement d'une page ne correspond pas obligatoirement au vidage d'une autre page.

Ces chargements/déchargements de page provoquent une activité importante et improductive (overhead) du disque et du processeur que les superviseurs tentent de limiter autant que possible. C'est dans ce but que ne seront vidées sur disque que les pages qui ont été modifiées en MC, modifiées depuis leur dernier chargement. D'où un nouvel indicateur dans les entrées de la TdP, appelé M (modifié, modified):

V	M	adresse

M=0 —> page non modifiée
M=1 —> page modifiée

Si l'indicateur V (valide) est géré par le logiciel (ou par des microprogrammes si le morceau correspondant du superviseur est microprogrammé), l'indicateur M est bien sûr géré par le matériel: toute instruction modifiant le contenu d'une page doit mettre M à un. Dans le VAX-11/780 ce bit M est recopié dans les RT et selon [CLA85] il y est mis à un de 2 à 12 fois pour 100 000 instructions exécutées, en fonction du type de charge.

Les applications peuvent elles-mêmes provoquer une pagination délibérée afin d'aider le système à éliminer les pages inutiles: rôle des macros PGOUT et PGLOAD sous MVS en mode superviseur, en mode problème l'utilisateur peut libérer des cases, sur mémoire auxiliaire.

Pour les programmes qui ont des contraintes de performance, il est possible d'inhiber la traduction d'adresse. On parlera alors de zone V=R, par opposition à la zone soumise à la pagination et appelée V=V. Dans la zone V=R l'adresse virtuelle est égale à l'adresse réelle. Un autre moyen d'améliorer les performances est de bloquer la page en mémoire afin qu'elle ne soit pas vidée sur mémoire secondaire, qu'elle évite la pagination: la primitive VMS $LCKPAG fixe ainsi une page, $ULKPAG la libère.

En résumé, la pagination dynamique présente comme **avantages**:
- une grande zone (espace d'adresses) adressable par les formes exécutables. Elle peut avoir une taille de loin supérieure à celle de la mémoire centrale;
- la mémoire centrale est utilisée avec plus d'efficacité: le code non utilisé n'est pas chargé;
- le degré de multiprogrammation peut encore être accru puisque un plus grand nombre de processus pourra être chargé en mémoire centrale, d'où un débit global plus important si le système est bien réglé (tune).

Quant aux **inconvénients**:
- on retrouve la majeure partie de ceux vus pour la pagination statique;
- la taille des tables est accrue (le nombre de formes exécutables chargées est accru et leurs espaces adressables sont plus grands);
- il y a un risque d'**emballement** (dérapage, thrashing en américain) le système passe de plus en plus de temps à gérer les fautes de page, allant jusqu'à leur consacrer plus de temps qu'à l'exécution proprement dite des formes exécutables si la charge est excessive. L'emballement est dû à deux causes: un trop grand nombre de processus et des processus trop grands. Les solutions en sont la diminution du nombre de processus, par vidage et/ou ralentissement de la planification afin de réduire le taux de multiprogrammation; pour ce qui concerne les processus trop gros, la technique de WS (Working Set) apporte une solution. [IBMPE] indique ainsi que 10 fautes de page par seconde prennent 1 % de la puissance processeur d'un 3033, le transfert des 10 pages de 4 Ko sur un 3350 charge le canal à moins de 6 % (5,7 ms par page). [SCH80] donne des ordres de grandeur acceptables de 20 fautes de page par seconde sur un 3031 ou un 158, de 30 sur un 168 ou un 3032, de 50 sur un 3033.

Un des effets de la pagination peut être illustré par un exemple tiré de [REI73] qui donne les temps d'exécution d'un programme Cobol de 64 Ko (37 de code, 27 de tampons) dans divers environnements (tableau 11u). Ce programme est utilisé sous deux formes: une forme dite "non optimisée", et une forme "optimisée" par regroupement des modules les plus actifs, ce qui fait baisser le nombre de fautes de page. La différence entre MFT, système sans pagination, et V=R sous VS1 correspond à une déperdition (overhead) accrue entre MFT et VS1: d'une façon générale l'évolution d'un logiciel peut conduire à une amélioration de performance, de qualité, ou encore à un accroissement des fonctionnalités; dans ce dernier cas il est très fréquent de voir le logiciel devenir plus gros en taille et plus gourmand en processeur. La différence entre V=R et R=70 (Ko) correspond à la mise en oeuvre des mécanismes de pagination d'entrée et à la traduction des adresses: 21 fautes correspondent au chargement de 80 Ko, chaque page faisant 4 Ko; il n'y a pas de pagination de sortie (page-out) puisque la taille mémoire est suffisante pour contenir tout le programme une fois chargé. On constate que le programme optimisé est d'autant meilleur, relativement, que la taille mémoire allouée est plus petite.

NON OPTIMISE SE	TEMPS PC	TEMPS ECOULE	DEFAUTS ENTREE	DE PAGE SORTIE
MFT	80	2,90	-	-
V=R	83	3,45	21	0
R=70	91	3,48	530	307
R=60	93	4,31	1675	1240
R=50	108	9,88	12604	10607
OPTIMISE				
MFT	80	2,88	-	-
V=R	82	2,85	17	0
R=70	91	3,16	472	303
R=60	92	3,68	1062	1363
R=50	102	6,77	6106	8494

Les lignes "V=" ou "R=" correspondent à l'exécution sous OS/VS1.

TEMPS PC: temps processeur central.

Tableau 11u: effet de la pagination sur un programme Cobol de 64 kilo-octets.

Un autre effet classique de la pagination est illustré par le traitement des tableaux. Considérons les deux versions A et B du programme Fortran ci-dessous dans lequel le rôle des lignes et des colonnes a été inversé:

```
A       INTEGER X(200,500)
        DO 100 J=1,500
        DO 100 I=1,200
    100 X(I,J)=0
        STOP
        END
```

```
B   ...
        DO 100 I=1,200
        DO 100 J=1,500
        ...
```

Sous OS/VS1 tournant sur un 370/145 avec 256 Ko de mémoire centrale dont 150 Ko disponibles pour les travaux (job), avec une partition de 512 Ko en mémoire virtuelle, on obtient les résultats suivants:

	TEMPS PC	TEMPS ECOULE	PAGES TRANSFEREES EN ENTREE	SORTIE	
PROGRAMME A	2,95	13,85	49	184	(temps en
PROGRAMME B	50,85	2410,11	38655	38793	secondes)

[BAR82] donne dans un contexte TSO avec 20 utilisateurs, un temps de résolution des défauts de page en millisecondes de:

	MVS/SP1.1	SP1.3
4341-2 avec 4 Mo	47	56
4341-2 avec 8 Mo	29	31

Sperry (Unisys aujourd'hui) parle de **pagination logique** dans le cas de l'OS1100: la 1100 a un mécanisme de base séparant le code et les données qui conduit à implanter deux partitions translatables (deux "bancs") en mémoire pour chaque programme. Ce mécanisme a été généralisé à plus de deux bancs, en distinguant des bancs "statiques" résidents en mémoire centrale pendant toute l'exécution du programme et des bancs "dynamiques" chargés à la demande. Selon le modèle de 1100 il y aura plus ou moins de registres qui permettront d'accéder aux bancs en mémoire: Sperry parle de "fenêtre" (window), de "multi-fenêtrage" pour traduire le fait que plusieurs bancs sont directement accessibles (par les registres en question) à un instant donné; l'ensemble de ces bancs constitue effectivement une fenêtre sur l'ensemble du programme, le programme pouvant être plus grand que l'espace directement adressable. Ces bancs ressemblent donc à des segments et la "pagination logique" de l'OS1100 n'est pas une pagination.

Le HP1000 est à l'origine un mini 16 bits avec un espace adressable de 64 Ko; avec RTE-A l'option CDS (Code and Data Separated) permet de séparer code et données. La gestion mémoire est basée sur un système de partitions qui ne relève pas de la pagination, mais à l'intérieur d'une partition il est possible de charger le code par "segments", un programme pouvant contenir jusqu'à 128 segments de 1 à 31 pages de 2 Ko: la zone code de la partition est découpée en n blocs de longueur identique, égale à celle du plus grand segment à charger; n segments au plus peuvent ainsi résider simultanément dans la partition. L'option VMA offre un espace de 64 Kpages pour les données et correspond à la pagination dynamique.

11.8.1 – Pré-pagination.

Quand se produit une faute de page, il y a trois décisions à prendre :
- appliquer la règle de remplacement : **quelle** page vider?
- choisir **quand** il faut charger la page absente,
- choisir le cadre **où** placer cette page absente.

Une faute de page conduit au chargement d'au moins une page. En se basant sur le principe de la localité des références, certains systèmes (ou installations) estiment qu'il y a une forte probabilité pour que les (n-1) pages suivantes soient sous peu référencées. Sur une faute de page, ils ne chargent pas seulement la page référencée, mais

les "n-1" pages qui la suivent en mémoire virtuelle (paramètre PFGDEFAULT de VAX/VMS). Il y a ainsi une seule E/S physique et une seule activation du module de gestion des pages pour n pages. C'est ce que l'on appelle la **pré-pagination** (prepage, prefetch). Dans GCOS8 on trouve le concept de "groupe de pages" (page group), dans VAX/VMS les pages pré-chargées sont appelées "cluster". On trouve aussi un mécanisme de prépagination dans Primos. Sur Guardian de Tandem, deux pages consécutives sont systématiquement chargées.

La pré-pagination ne s'exerce pas toujours en prenant "n" pages consécutives, le choix des "n" pages peut relever d'algorithmes plus sophistiqués.

Notons que le concept de pré-pagination est tout-à-fait analogue à celui de bloc dans les antémémoires: lorsqu'un mot n'est pas trouvé dans l'antémémoire, on charge automatiquement les (m-1) autres mots du bloc comme nous l'avons décrit dans le tome I.

Toujours dans le but d'optimiser le coût de la pagination d'entrée, certains sytèmes accumulent un nombre prédéfini de demandes avant de lancer l'E/S qui chargera les pages. Le programme canal est bien sûr trié en fonction des adresses sur disque, d'où un gain global sur le temps d'exécution du programme canal. Ceci peut provoquer une certaine dégradation des temps de réponse puisque les premières demandes sont mises en attente.

11.8.2 – Post-pagination.

En ce qui concerne la pagination de sortie, certains systèmes (VAX/VMS par exemple) ne vident pas immédiatement les pages sur disque. Ils accumulent un certain nombre de pages à vider et ne les écrivent sur disque que lorsqu'un certain seuil est atteint:
- n demandes ont déjà été enregistrées,
- le nombre de cadres libres (disponibles) en MC pour y ranger des pages atteint une limite basse.

On nomme ces techniques **post-pagination**. Il peut arriver ainsi qu'une faute de page se produise pour une page en attente d'être vidée; la faute de page ne conduit dans ce cas à aucune E/S, mais à la réinsertion de la page dans la liste des pages actives. On parlera alors de **faute de page logique**.

Dans cet ordre d'idée VAX/VMS distingue:
- les fautes de page rapides ("fast page fault") lorsque la page est encore en MC, synonymes de faute de page logique;
- les fautes de page lentes (slow page fault) lorsqu'une entrée/sortie est nécessaire sur disque.

L'expression "vidage sur disque" ne doit pas être prise au sens restrictif, mais plutôt comme "vidage sur mémoire secondaire". Si le disque est la mémoire secondaire type actuellement, on utilise aussi des tambours et des mémoires à semi-conducteurs. Le 3090 d'IBM dispose ainsi d'une mémoire étendue (expanded storage, mémoire d'arrière plan chez IBM-France, cf. [TUC86A]) dont la case est une page de 4 Ko; le transfert d'une page de/vers la mémoire centrale se fait de façon synchrone, c'est-à-dire en bloquant le processeur, pendant 75 microsecondes, l'action du SE comprise, sur un 3090/400. La lecture en

MC est faite par microprogrammes, le vidage par logiciel ([DUB86]).

11.8.3 – Choix de la page à vider.

Il existe de nombreux algorithmes (page removal, page replacement algorithm) utilisés pour choisir la (les) page(s) à vider sur mémoire virtuelle, pratiquement un par système:

- **aléatoire** (random): la page est choisie au hasard. Contraire au principe de localité cet algorithme est peu utilisé mais s'est cependant révélé intéressant dans certains contextes transactionnels;
- FIFO (First In, First Out, premier entré, premier sorti): les pages sont ordonnées selon leur séquence de chargement, la première chargée sera la première remplacée. On choisit donc la plus ancienne. Le logiciel gère pour cela une liste des pages en MC;
- FINUFO (First In Not Used, First Out, premier entré non utilisé, premier sorti, LAM pour Look-Aside Memory) avec liste circulaire: comme pour FIFO le système gère une liste des pages, avec pour chaque page un bit indiquant si elle a été ou non utilisée (bit U ci-dessous). Un pointeur indique la prochaine page à vider et tourne dans la liste circulaire. Quand une faute de page arrive, on vide la première page non utilisée qui suit la position courante du pointeur. Cet algorithme a été utilisé dans Multics et dans CP-67 (ancêtre de VM);
- LFU (Least Frequently Used, le moins fréquemment utilisé): il fait l'hypothèse que les pages récemment utilisées le seront encore dans le futur proche. Cette méthode a le défaut de désavantager les pages récemment chargées si l'on n'y prend garde;
- LRU (Least Recently Used, le moins récemment utilisé) est d'usage fréquent. Le pur LRU est rarement utilisé car il conduit à modifier l'ordre des pages à chaque référence (le STAR 100 de CDC l'applique cependant pour les 16 pages les plus récentes). La majorité des systèmes se contente de l'indicateur appelé U (utilisé, used) ou R (référence, referenced) selon les machines, réservé à cet effet dans la table des pages:

V	M	U	adresse

U = 1 ——> page référencée
U = 0 ——> page non référencée

Périodiquement les indicateurs U sont remis à zéro, et lorsqu'il faut faire de la place en mémoire centrale, les pages avec U à zéro sont les premières à être vidées. Certains systèmes, comme MVS, ont plusieurs bits dans le champ U et peuvent ainsi affiner l'algorithme, le champ étant incrémenté à chaque référence. Notons que le VAX ne dispose pas de cet indicateur sous VMS, il a cependant pu être simulé sur le VAX sous Unix.

Illustrons ces mécanismes par l'exemple suivant dans lequel on dispose d'une mémoire centrale de 3 cadres, pouvant donc contenir 3 pages. Les références de page à page se font dans la séquence suivante:

4 3 2 1 4 3 5 4 3 2 1 5

Le contenu de la mémoire centrale évoluera comme suit si l'on utilise un algorithme FIFO:

instant	page référencée	contenu cadres MC 1	2	3	faute de page
1	4	4			oui
2	3	4	3		oui
3	2	4	3	2	oui
4	1	1	3	2	oui
5	4	1	4	2	oui
6	3	1	4	3	oui
7	5	5	4	3	oui
8	4	5	4	3	
9	3	5	4	3	
10	2	5	2	3	oui
11	1	5	2	1	oui
12	5	5	2	1	

On aboutit ainsi à 9 fautes de page, soit une fréquence de 9/12: 75 % puisqu'il y a eu 12 références de pages différentes. L'algorithme FIFO a deux défauts importants:

- si une page est très utilisée, elle finira malgré tout par devenir la plus vieille et sera donc vidée de la mémoire centrale, même si elle doit être référencée juste après;
- il est sensible à certains effets de bord.

D'autres algorithmes se sont révélés plus efficaces. En reprenant l'exemple ci-dessus, mais avec une mémoire de 4 cadres, on obtient:

instant	page référencée	contenu cadres MC 1	2	3	4	faute de page
1	4	4				oui
2	3	4	3			oui
3	2	4	3	2		oui
4	1	4	3	2	1	oui
5	4	4	3	2	1	
6	3	4	3	2	1	
7	5	5	3	2	1	oui
8	4	5	4	2	1	oui
9	3	5	4	3	1	oui
10	2	5	4	3	2	oui
11	1	1	4	3	2	oui
12	5	1	5	3	2	oui

On passe ainsi à 10 fautes de page au lieu de 9 précédemment, ce qui illustre une anomalie de l'algorithme FIFO: une mémoire de 4 cadres provoque plus de fautes de page qu'une mémoire de 3 cadres.

Passons maintenant à un algorithme de type LRU qui, par définition, videra la page dont la dernière référence remonte le plus loin dans le temps (FIFO par contre vide la page qui a résidé le plus longtemps en mémoire centrale, qu'elle ait ou non été peu ou prou

référencée). Sur le tableau ci-dessous nous représentons simultanément les deux mémoires centrales, celle avec 3 cadres et celle avec 4 cadres:

instant	page référencée	MC 3 cadres				MC 4 cadres				
		1	2	3	fp	1	2	3	4	fp
1	4	4			o	4				o
2	3	4	3		o	4	3			o
3	2	4	3	2	o	4	3	2		o
4	1	1	3	2	o	4	3	2	1	o
5	4	1	4	2	o	4	3	2	1	
6	3	1	4	3	o	4	3	2	1	
7	5	5	4	3	o	4	3	5	1	o
8	4	5	4	3		4	3	5	1	
9	3	5	4	3		4	3	5	1	
10	2	2	4	3	o	4	3	5	2	o
11	1	2	1	3	o	4	3	1	2	o
12	5	2	1	5	o	5	3	1	2	o

fp: faute de page

On obtient 10 fautes de page pour la mémoire centrale de 3 cadres, soit pour cet exemple-ci, un résultat moins bon que FIFO. Avec la mémoire de 4 cadres, on tombe à 8 fautes de page, contre 10 pour FIFO. LRU dispose d'une propriété importante appelée "propriété d'inclusion". Soit:

P: la séquence des pages référencées,
n: le nombre de cadres de la mémoire centrale,
t: l'instant d'observation,
M(P,n,t): l'ensemble des pages en MC à l'instant t,

on démontre que l'on a:

$$M(P,n,t) < M(P,n+1,t)$$

c'est-à-dire que l'ensemble M(P,n+1,t) inclut l'ensemble M(P,n,t), ce que vérifie l'exemple ci-dessus. Ainsi, l'augmentation de la taille de la MC ne peut pas provoquer l'accroissement du nombre de fautes de page, effet mis en évidence avec FIFO.

11.8.4 – Seuils, vidage et pagination.

Certains SE gèrent des seuils au-delà desquels des processus sont vidés sur mémoire secondaire ou ramenés en mémoire centrale (swapping). Ces seuils sont généralement basés sur le nombre de cadres disponibles en mémoire centrale:

- quand il est trop bas le SE vide des processus,
- quand il est suffisamment grand, le SE ramène des processus.

On trouve ceci dans VAX/VMS, MVS entre autres.

Ces cadres libres assouplissent le fonctionnement et permettent de découpler l'entrée des pages en mémoire centrale (page-in) de leur sortie (page-out).

Il peut cependant advenir que la pagination s'emballe et n'offre pas de solution à cet emballement car les processus ont un comportement qui échappe aux efforts régulateurs des algorithmes du système.

Il faut alors compléter la pagination par le vidage (swapping) des processus, technique présentée plus haut pour les systèmes sans pagination. Nous illustrerons cette technique par deux exemples.

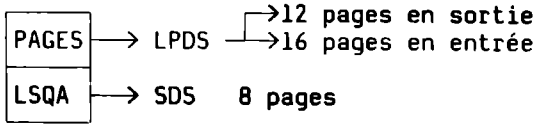
Dès sa première version **VMS** a comporté pagination et vidage; un troisième niveau intermédiaire appelé "trimming" (réduction) a été ajouté par la suite. Il correspond à un vidage partiel du WS (cf. plus loin). L'ensemble des processus en mémoire centrale (les processus non vidés) à un moment donné est appelé "balance set": quand un processus entre dans cet ensemble, il reçoit un quantum de temps processeur qu'il devra normalement épuiser avant d'être candidat au vidage. Il ne faut pas confondre ce quantum avec celui du distributeur.

Pour la pagination et le vidage, **MVS** a dans un premier temps utilisé un seul type de fichier appelé PDS (Page Data Set, fichier des pages), celui de la pagination. LPDS (Local PDS) y désigne les fichiers de page utilisés pour vider les pages locales aux espaces d'adresses (cf. chapitre 12). [FRI81A] donne l'exemple ci-dessous, dans lequel le vidage d'un AS, puis son retour en mémoire centrale, provoquent un transfert global de 44 pages. Les pages y sont classées selon deux critères, l'un concerne le fait qu'elles ont été référencées (R) ou non (NR), l'autre est associé à leur modification (M) ou non modification (NM) pendant leur résidence en mémoire centrale:

TYPE DE PAGE	NOMBRE DE PAGES	PAGES TRANSFEREES EN SORTIE	ENTREE
NRNM	4	0	0
NRM	4	4	0
RNM	8	0	8
RM	8	8	8
LSQA	8	8	8

une première amélioration (MVS/SP release 1) a conduit à écrire les pages dans des cases contiguës (contiguous slot allocation) sur disque afin d'améliorer les temps d'entrée/sortie. Les pages étaient auparavant écrites dans des cases individuelles, d'où de nombreux mouvements de bras et délais rotationnels (cf. tome IV), même si leurs transferts étaient regroupés dans un même programme canal. Avec la version 1.3 de MVS/SP (release 3) on en arrive à écrire les pages référencées non modifiées, afin de simplifier l'algorithme global (figure 11v).

Une autre évolution de MVS a introduit les fichiers de vidage (SDS, Swap Data Set) qui avaient l'avantage de décharger les fichiers de pagination (les PDS), sollicités par ailleurs pour la pagination, ne l'oublions pas. Dans un premier temps seule la zone LSQA a été vidée dans un fichier de vidage:



les fichiers de vidage (SDS) permettent de soulager les fichiers de pagination (PDS) qui passent dans cet exemple de 44 à 28 pages transférées.

Une autre amélioration des performances apportée par MVS/SP R3, appelée "vidage étendu" (extended swap) conduit à généraliser les fichiers de vidage à la grande majorité des pages de l'espace d'adresses:

NRNM	0	0
NRM	LPDS	0
RNM	SDS	SDS
RM	SDS	SDS
LSQA	SDS	SDS

le vidage étendu optimise le fonctionnement global dans la mesure où il permet de diminuer les déplacements de bras sur le disque. Un bit est ajouté pour les pages de type RM si elles ont été modifiées depuis leur dernière écriture dans le fichier des pages (PDS) afin de pouvoir l'y écrire en cas de "vol". Pour donner une idée des améliorations apportées par le vidage étendu [BER81] présente des résultats obtenus sur un 3033 de 8 Mo, avec des disques 3350 sous MVS/SE, pour une charge de travail purement interactive (TSO); malgré un plus grand nombre de pages transférées, les temps de réponse sont meilleurs. On constate d'ailleurs que le nombre d'E/S physiques est moindre, et leur durée l'est certainement aussi:

NBR COMMANDES TSO/SECONDE	4,2	5,9	6,6	7,5
TEMPS DE REPONSE SANS VE	1,1	1,3	1,6	2,3
AVEC VE	0,8	1,0	1,2	1,3
PAGINATION TOTALE/SECONDE				
AVEC VE	164	232	262	318
SANS VE	218	310	362	394
NBR SIO/SECONDE DUES A LA				
PAGINATION AVEC VE	39	52	55	60
SANS VE	33	45	50	55

VE: vidage étendu

Ces exemples auront permis d'illustrer combien est complexe la gestion des mémoires paginées. Ils soulignent aussi l'évolution constante des algorithmes, les constructeurs cherchant à améliorer les résultats au travers des différentes versions des logiciels. Ils illustrent enfin le type de problème qui se pose et les solutions apportées dans des SE réels.

NRNM	0	0	0	0
NRM	4	0	4	0
RNM	0	8	bloc	bloc
RM	bloc	bloc	bloc	bloc
LSQA	bloc	bloc	bloc	bloc
	MVS/SP R1		MVS/SP R3	

Figure 11v:
avec l'allocation contiguë des cases (slot) sur disque, les pages sont regroupées en blocs qui sont stockés de façon contiguë sur disque.

11.8.5 – Working set (WS).

Vis-à-vis des différents processus ou groupes de processus en compétition pour obtenir de la place en mémoire centrale, on distingue deux grands types d'algorithme de choix de la page à vider:

- ceux basés sur un choix **global**: quand une faute de page se produit du fait du processus Pi, toute page en mémoire centrale est susceptible d'être vidée; MVS parlera de **vol de page** (page stealing) lorsqu'il n'y a plus asssez de cadres libres et qu'il faut prendre un cadre occupé par une page d'un espace d'adresses. L'expression "pagination contre le système" correspond au même type d'algorithme;
- ceux basés sur un choix **local**: la page vidée sera choisie parmi les pages en MC du processus fautif Pi, afin d'éviter qu'un processus gourmand en place MC, ne pénalise exagérément les autres processus. C'est le cas du VAX sous VMS: au niveau de chaque processus il utilise un algorithme FIFO. L'expression "pagination contre le processus" se réfère à ce type de pagination. Les algorithmes de WS (Working Set) présentés dans cette section en font partie.

Lorsque nous avons évoqué l'emballement (thrashing) plus haut, nous avons cité deux causes principales: le trop grand nombre de processus, la trop grande taille de certains processus. La notion de WS que nous introduisons ici est une réponse à la seconde cause d'emballement, elle a pour but de cerner les besoins réels des processus pour ce qui concerne la taille mémoire, c'est-à-dire leur en donner suffisamment pour qu'ils ne perdent pas trop de temps à paginer, tout en les empêchant d'en prendre trop afin de ne pas perturber les autres processus. Il convient bien entendu de garder dans le WS les pages les plus utiles à un instant donné.

Nous définirons l'**«ensemble de travail»** ou **«espace vital»** (working set abrégé en WS) comme le nombre de cadres nécessaires à un processus pour qu'il s'exécute sans causer un nombre excessif de fautes de page. Denning définit le WS d'un programme à l'instant t comme l'ensemble des pages référencées pendant l'intervalle [t-T+1,t], T étant un paramètre appelé "fenêtre". Denning stipule que toutes les pages du WS courant doivent rester en MC et que les autres peuvent être mises sur mémoire secondaire. De nombreuses mesures du nombre de fautes de page en fonction de la taille mémoire allouée, conduisent à une courbe qui évolue de façon exponentielle, la taille du WS correspond au début du coude exponentiel de cette courbe.

VAX/VMS implémente cette notion (visible de façon externe par la commande DCL SHOW WORKING_SET par exemple) en définissant pour chaque utilisateur et pour chaque forme exécutable (une "image" dans la terminologie DEC) un nombre précis de cadres à lui allouer de façon sûre.

Sous GCOS7 on trouvera le terme de DWS (Declared WS), cette donnée est utilisée par le planificateur. Multics et GCOS8 implémentent aussi la notion de WS. Dans GCOS7 comme dans MVS, l'algorithme peut être global, malgré le terme de WS et tous les emplois de l'expression "working set" ne correspondent pas à cette définition. Ainsi dans MVS elle correspond à la taille occupée par un AS en mémoire centrale mais ne signifie pas que l'algorithme de pagination est local; il y a dans

MVS, depuis la version 2.2, une sorte de WS inversé pour les AS qui ont l'option d'«isolation en mémoire» dans la mesure où ils sont protégés contre les excès de l'algorithme global de MVS. Auparavant un des objectifs internes était de maintenir le taux de pagination globale à un niveau raisonnable, chaque AS étant traité de façon égalitaire. Le WS d'une application est défini comme l'ensemble des cadres locaux qui lui sont alloués. Le WS commun correspond à l'ensemble des cadres alloués à la mémoire commune. De façon régulière, MVS calcule le produit WS*temps et le compare au produit défini dans les objectifs de performance de l'installation ("target" WS*temps des IPS); tant que le produit est inférieur au produit prédéfini, il n'y a pas de vol de page par SRM pour l'espace d'adresses considéré, sauf si tous les AS sont en dessous du produit prédéfini. A la taille MC peuvent aussi être associés des seuils mini-maxi de taux de pagination d'entrée (page-in rate), qui permettent à SRM de calculer le "target WS" qui doit lui aussi se trouver entre deux valeurs mini/maxi prédéfinies. SRM jouera sur le taux de multiprogrammation (MPL) pour satisfaire les objectifs. Ceci n'est donc pas un algorithme local tel que nous l'avons présenté plus haut, il ne conduit pas à paginer à l'intérieur des processus, mais à protéger certains ensembles contre la pagination des autres.

11.8.6 – Pagination à deux niveaux ou plus.

Sur IBM 370 par exemple, un processus peut adresser 16 Mo (adressage dit à 24 bits). Avec des pages de 4 Ko, la table des pages représente au maximum 16 Ko contigus à raison de 4 octets par entrée (PTE avec 12 bits pour le numéro de cadre). Sur une machine 370 avec 20 processus en cours d'exécution, il faut ainsi jusqu'à 320 Ko de mémoire pour les seules tables de page, avec des contraintes de contiguïté de 16 Ko et de résidence car elles ne doivent pas être sujettes à la pagination.

Sur le VAX un processus peut adresser jusqu'à environ 1 Go de mémoire virtuelle privée, d'où une table des pages de 8 Mo par processus (pages de 512 octets, 4 octets par entrée).

Pour surmonter ces problèmes posés par:
- la zone contiguë importante requise par la table des pages,
- la non paginabilité de la table des pages,

certains systèmes offrent un second niveau de pagination avec un second niveau de pages appelées **hyperpages**. Un groupe de n pages consécutives en mémoire virtuelle forme une hyperpage (appelée parfois de façon abusive segment, segmentation linéaire). L'adresse virtuelle prend alors la forme tri-dimensionnelle suivante:

h	p	d

h: numéro de l'hyperpage (8 bits sur 370)
p: numéro de la page (4 bits sur 370)
d: déplacement dans la page (12 bits sur 370)

Seule la table des hyperpages doit être contiguë et résidente en MC. Dans les 370 le registre de contrôle CR1 contient l'adresse de la table des hyperpages; il ne change quand un nouvel AS prend la main,

les entrées propres à l'AS sont modifiées en conséquence. En ce qui concerne les RT, un défaut de RT peut alors conduire à deux références MC (double miss), une par niveau de table si aucun RT ne contient la traduction d'adresse pour la table des pages.

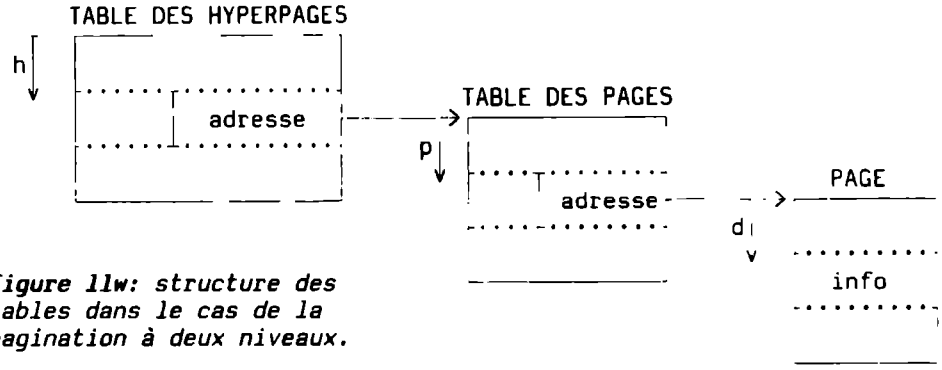


Figure 11w: structure des tables dans le cas de la pagination à deux niveaux.

Les mesures de [CLA85] sur un VAX-11/780 avec une charge interactive indiquent que:

- 3 % des références conduisent à des échecs doubles (double miss), le numéro de cadre de la page n'est pas dans les RT, le numéro de l'hyperpage non plus;
- 42 % des échecs simples concernent la région système (voir chapitre 12);
- 47 % sont dus à la région utilisateur P0;
- 8 % se font dans la région de contrôle P1, la meilleure localité y étant certainement due au fait que c'est dans cette région que sont installées les piles de chaque processus utilisateur.

Les registres de traduction doivent donc avoir le contenu ci-contre:

h	p	adresse physique

et s'ils peuvent être partagés par plusieurs processus simultanément:

j	h	p	adresse physique

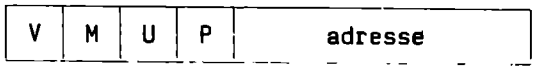
Le nombre de niveaux n'est pas limité à deux. Dans le 90X de Pyramid on peut avoir jusqu'à trois niveaux, seuls deux sont activés si le processus n'a pas besoin d'adresser plus de 16 Mo.

1 NIVEAU	2 NIVEAUX	3 NIVEAUX
.43xx d'IBM .VAX région système	.370, 33XX, 308X, 3090 .NS16000 .iAPX386 .GCOS8 .VAX région utilisateur	.Pyramid

Figure 11x: exemples de systèmes avec un à trois niveaux de pagination.

11.9 – Protection.

La pagination permet d'installer des éléments de protection au niveau de la page:



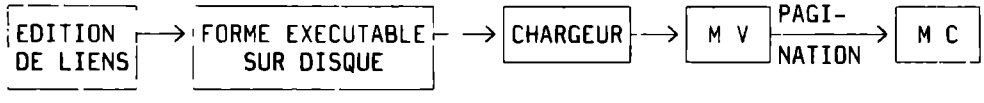
P: bit ou champ protection

Ces protections sont généralement liées aux opérations suivantes:
- lecture de la page
- exécution d'une instruction dans la page
- écriture dans la page
- suppression de la page
- changement des protections

Dans MVS/370 le bit "lecture seulement" est dans la table des hyperpages, il passe dans la table des pages dans MVS/XA. Lorsque la machine dispose d'anneaux, ces protections peuvent être précisées pour chacun des anneaux. Le chapitre 14 détaillera ce point.

11.10 – Chargement des programmes.

Le chargeur a pour fonction de charger une forme exécutable en mémoire: mémoire centrale dans un système classique, mémoire virtuelle dans un système avec ce type de mémoire. C'est ensuite le gestionnaire de la mémoire centrale qui chargera les pages en fonction des besoins:



11.11 – Exemples.

11.11.1 – Pagination sous DOS/VS.

La première version de DOS/VS implémente un algorithme dit de WS, au niveau du système tout entier, et non pas au niveau de chaque processus. Comme MVS, DOS emploie le terme WS avec un sens différent de celui que nous avons adopté, ce n'est donc pas un algorithme local ainsi que nous l'avons défini. Dans DOS/VS:
- un WS est l'ensemble des pages utilisées durant une période de temps que nous appellerons P;
- les pages non référencées pendant P sont notées dans une liste. Elles sont candidates (eligible) au remplacement et mises dans deux listes (queues) FIFO:
  * une liste des pages inchangées en mémoire centrale,
  * une liste des pages modifiées depuis leur chargement en MC;
- les premières pages remplacées sont celles qui sont inchangées. Quand leur liste est épuisée, on passe à la seconde.

Si la pagination est excessive, DOS/VS arrête l'exécution des tâches de plus basse priorité (d'abord BG, puis F4, etc.). DOS/VS a été conçu sans mécanisme de cadres libres avec seuils (page-out ahead) afin de simplifier au maximum l'algorithme de gestion des pages; sur une

machine moyenne (cas de DOS/VS) on évite de dépenser trop de temps pour la pagination. Notons que dans DOS/VS, comme dans VS1, il y a un seul fichier de pagination, chaque page y a une case associée prédéfinie ([DUB86]).

11.11.2 – Pagination sous MVS.

Le mécanisme de la pagination sous MVS est basé sur le bit R (Referenced) des clés de protection que nous détaillerons dans le chapitre 14:

0123	4	5	6	7
clé	F	R	C	r

F: Fetch protect R: Referenced
C: Changed r: reserved

A la différence de la PFT, les clés sont gérées directement par le matériel. Nous considérerons cependant ici que les bits R et C font fonctionnellement partie de la table des cadres (PFT) gérée par RSM (Real Storage Manager), le composant de MVS chargé de la gestion de la mémoire centrale:

# d'AS	# d'hyperpage	# de page	U/D	R	C	compteur UIC

: numéro
U/D: utilisé ou disponible

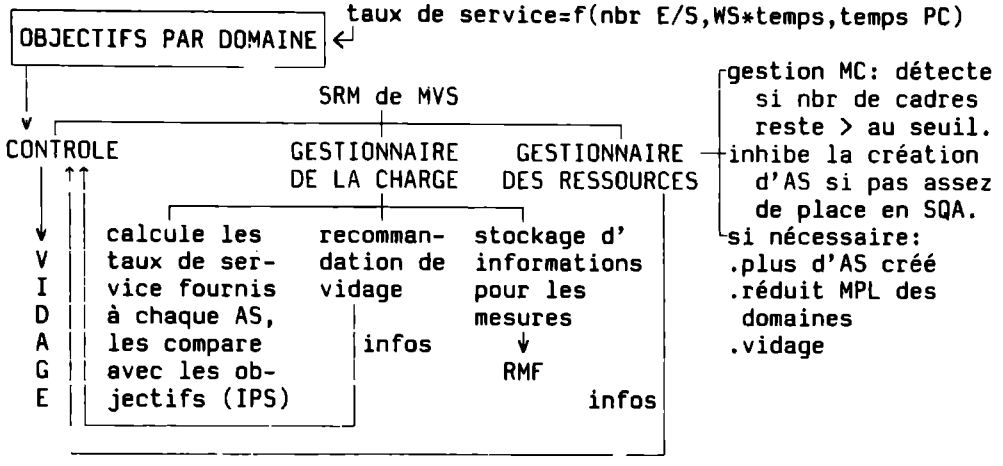
Les entrées de cette PFT ont 16 octets sur 370, 32 sur XA. En plus des numéros d'AS, d'hyperpage et de page représentés ci-dessus, chaque entrée de la PFT contient deux pointeurs avant et arrière vers d'autres entrées du même AS, ainsi qu'un compteur (Fixed Count) contenant le nombre de processus faisant des E/S sur ce cadre.

Périodiquement (toutes les secondes selon [BER81], variable d'une installation à une autre) RSM teste le bit R: s'il est égal à un, la page a été référencée pendant le dernier intervalle et R est remis à zéro, de même que le compteur UIC (Unreferenced Interval Count). S'il est égal à zéro, la page n'a pas été référencée pendant cet intervalle et le compteur est incrémenté de un. Les pages avec un compteur élevé sont candidates au vol de page et RSM maintient une valeur de référence de l'UIC appelée «UIC moyen du système»; elle témoigne du degré de contention pour la mémoire et règle indirectement la fréquence du vol de page: plus l'UIC est élevé, plus la fréquence sera faible.

Tableau 11y: exemple de table des cadres sous MVS.

#AS	#PAGE	U/D	COMPTEUR	R	C	
	1	0		1	1	page toujours référencée et modifiée
	1	0		1	0	page toujours référencée non modifiée
	1	5		0	0	cadre à récupérer en premier
	1	2		0	1	vider la page sur disque avant de récupérer le cadre (C=1)
	1	0		1	1	

Précisons que les clés ne sont pas vidées sur mémoire secondaire comme les pages mais sont sauvegardées en mémoire centrale ([DUB86]).



Le "vol" de page est déclenché par SRM qui avertit RSM dès qu'il y a trop peu de cadres libres (cf. [BAU85]). Dans le cas de MVS ces cadres disponibles font partie d'une liste (available frame queue qui constitue une des chaînes de la PFT); si sa taille passe en-dessous d'un seuil minimum (14 par défaut selon [BAU85]) SRM se met à "voler" des pages jusqu'à remonter à un seuil dit "ok" (18 par défaut):

1. en fait il ne se passe rien si ASM, le gestionnaire des mémoires auxiliaires (disques de pagination et de vidage), a suffisamment de pages en cours de vidage;
2. sur ordre de SRM, RSM prend des pages aux AS qui ont dépassé leur WS;
3. pour ce faire il lance l'algorithme LRU basé sur les UIC, avec comme critère de vol un UIC calculé par SRM en fonction des pages nécessaires;
4. le vol commence par les AS logiquement vidés, n pages sont volées chaque fois que le critère de vol est vérifié (UIC supérieur ou égal au critère);
5. le vol continue dans les zones privées puis dans la zone commune. Les AS "isolés en MC" (storage isolated AS) ne sont volés que s'ils ont plus de pages que leur WS objectif (target WS) et si le critère est vérifié. Depuis SP1.3 les AS ramenés en MC depuis moins de la moitié de la période de SRM ne sont pas sujets à ce vol afin de leur laisser le temps de s'installer;
6. si nécessaire le critère (UIC de vol) est diminué, et on reprend au point 4;
7. si nécessaire un algorihme dit de LRU "strict" est appliqué:
   7a. sur les AS logiquement vidés,
   7b. puis en ignorant les attributs d'isolation en mémoire,

RSM avertit SRM dès que le nombre de cadres libres dépasse un seuil

prédéfini, 18 par défaut. Il y a donc deux types d'activation du vol de page, l'un asynchrone par rapport aux AS, basé sur une temporisation qui déclenche SRM, un autre synchrone déclenché par un défaut de page dans un AS.

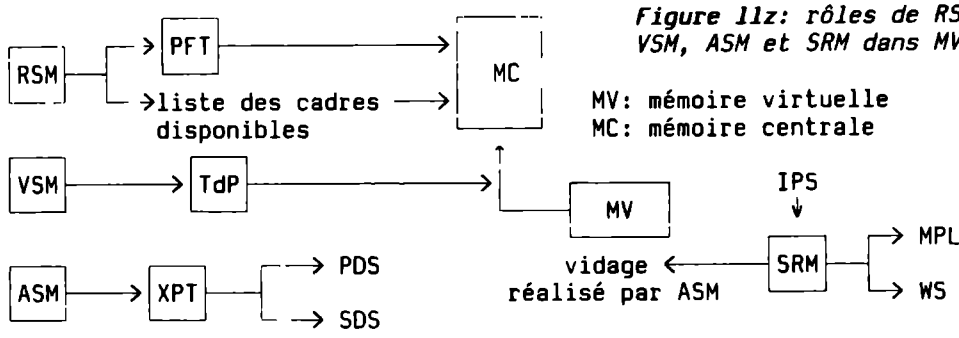
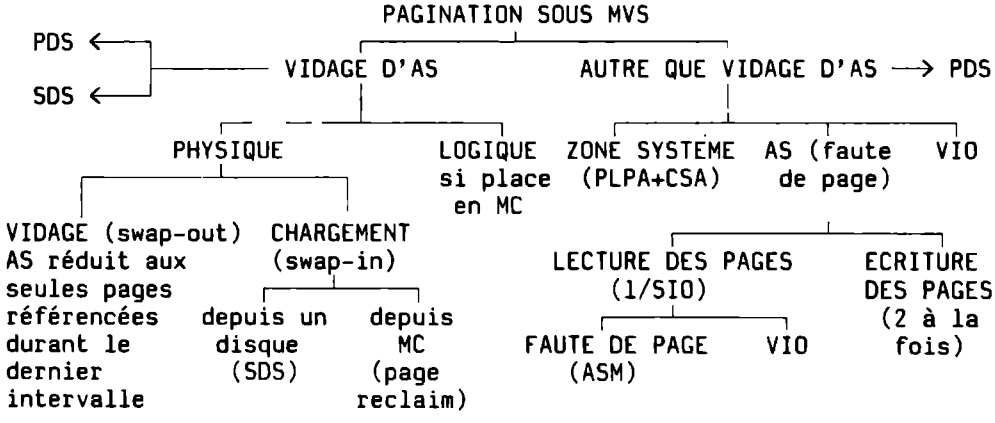


Figure 11z: rôles de RSM, VSM, ASM et SRM dans MVS.

Nous terminerons cette section avec un schéma tiré de [BER81] qui dresse un tableau des différents éléments de la pagination de MVS; le vidage logique correspond à un AS en MC avec l'état non exécutable:



11.11.3 – Pagination sous VAX/VMS.

La figure 11aa résume la carte de la mémoire centrale d'un VAX. L'«ensemble équilibré» (balance set) contient les processus actifs non vidés, appelés WS dans la terminologie VMS puisque cette technique de pagination y est utilisée. Les formes exécutables non chargées en mémoire virtuelle sont stockées dans des fichiers images avec le suffixe .EXE, les sections dont nous parlerons dans le chapitre 14 sont rangées dans les fichiers "sections". La pagination et le vidage disposent de leurs fichiers, les fichiers de page contenant les formes exécutables chargées en mémoire virtuelle.

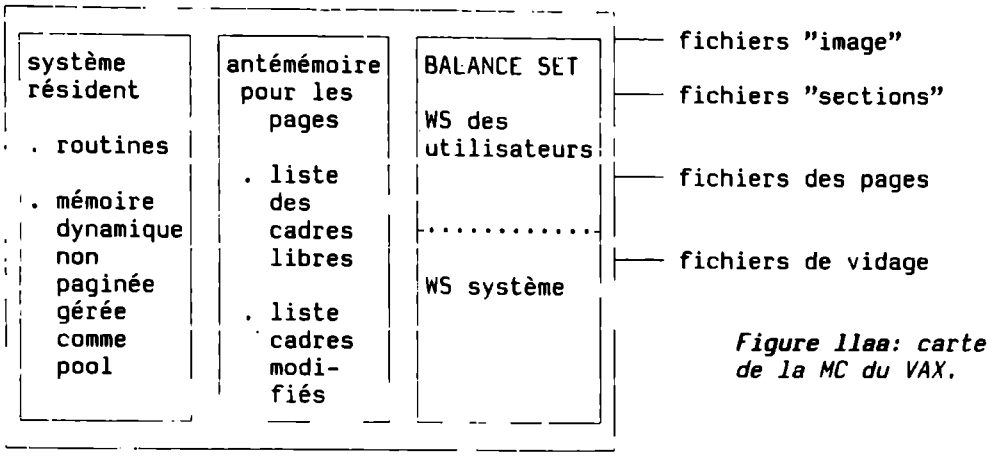
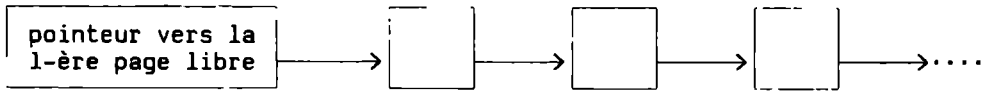


Figure 11aa: carte de la MC du VAX.

Au démarrage du système, la partie de la mémoire centrale attribuée à la pagination est structurée en un ensemble de cadres libres (free pages) chaînés:



Selon [VMS84E], en régime de croisière, la mémoire centrale est gérée par un triple mécanisme:
- d'ajustage automatique des WS, appelé AWSA (Automatic WS Adjustment); il intervient avec une fréquence définie par le paramètre AWSTIME (200 ms par défaut dans VMS 3.0);
- de réduction (trimming) des WS;
- de vidage en dernier recours.

Chaque fois qu'un processus a consommé au moins AWSTIME de temps processeur, ses besoins mémoire sont examinés. Il démarre avec une taille mémoire égale à WSDEFAULT telle que définie dans le fichier des autorisations ou fixée par la commande $ SET WORKING_SET ou grâce à la routine de service $ ADJUSTWSL. Si le processus pagine trop (taux supérieur à la limite PFRATH) il pourra recevoir un supplément de mémoire égal à WSINC cadres s'il y a encore GROWLIM cadres disponibles; il ne pourra cependant pas dépasser la taille WSQUOTA s'il n'y a pas au moins BORROWLIM cadres disponibles. Si la mémoire est peu sollicitée, le processus atteindre la taille WSEXTENT, mais sans jamais dépasser WSMAX définie pour tous les processus. Le système permet ainsi à un processus de se "répandre" lorsqu'il y a de la mémoire disponible et de ne pas paginer "bêtement" dans son WS.

Si le taux de pagination du processus est inférieur à PFRATL, le processus sera par contre susceptible de perdre des cadres en faveur d'autres WS; il en perdra WSDEC à chaque examen, avec une fréquence définie par AWSTIME, mais sans jamais descendre en-dessous de la

limite AWSMIN définie au niveau du système tout entier, c'est-à-dire pour tous les processus.

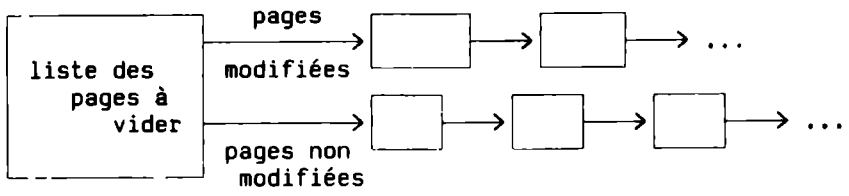


Figure 11ab: double liste d'attente des pages à vider sous VMS.

Ceci correspond au mécanisme AWSA auquel vient s'ajouter celui de réduction lorsqu'il n'est pas suffisant. La réduction se déclenche lorsque le nombre de cadres libres est inférieur à FREELIM. VMS essaie alors d'amener le nombre de cadres libres à la valeur FREEGOAL au moins et, s'il existe au moins MPW_THRESH pages dans la liste des pages modifiées, elles seront vidées sur disque et les cadres correspondants seront versés dans la liste des cadres disponibles. Si ceci ne permet pas d'atteindre FREEGOAL cadres disponibles, VMS réduit les processus en commençant par ceux de taille supérieure à WSQUOTA pour les ramener à WSQUOTA. Si ce n'est pas suffisant un second niveau de réduction est lancé et les processus sont ramenés à la taille SWPOUTPGCNT. Si ceci ne suffit pas, le troisième mécanisme, le vidage, entre en jeu pour vider en premier lieu les processus dans l'état suspendu, puis ceux dans l'état inactif (idle) depuis LONGWAIT, etc.

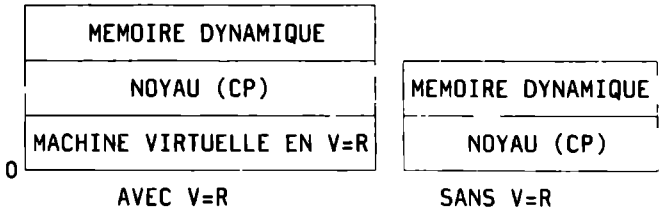
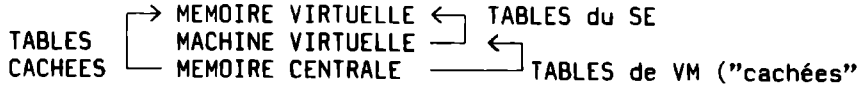


Figure 11ac: carte MC de VM avec/sans machine virtuelle en V=R.

11.11.4 – Pagination sous VM.

Pour localiser une page en mémoire centrale il faudrait utiliser les tables gérées par VM puis celles du SE concerné. Pour diminuer la déperdition impliquée VM utilise des tables de pages "cachées" (shadow table) qui pointent directement sur les cadres dans lesquelles les pages sont stockées, une par AS dans le cas de MVS.



Afin d'obtenir de meilleures performances, une machine virtuelle, MVS ou autre, peut s'exécuter en V=R, ce qui supprime un niveau de

pagination, celui de VM, puisque VM utilise alors la table des pages proposée (consciemment ou non) par le SE; il n'y a plus de table cachée. On peut de même dédier des canaux et des périphériques, d'où des gains au niveau de l'absolutisation des adresses dans les programmes canal (cf. chapitre 6).

Dans le cas de MVS il est possible d'utiliser un algorithme de type WS pour améliorer les performances (commande SET FAVOR). MVS est assuré de disposer d'un nombre de cadres "privés" qui est réservé à son seul usage, les pages correspondantes ne sont cependant pas fixées (locked) en MC.

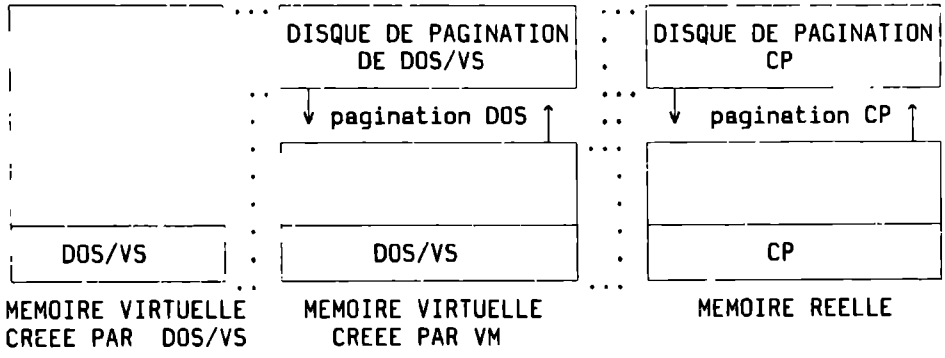


Figure 11ad: les 3 niveaux de mémoire avec VM.

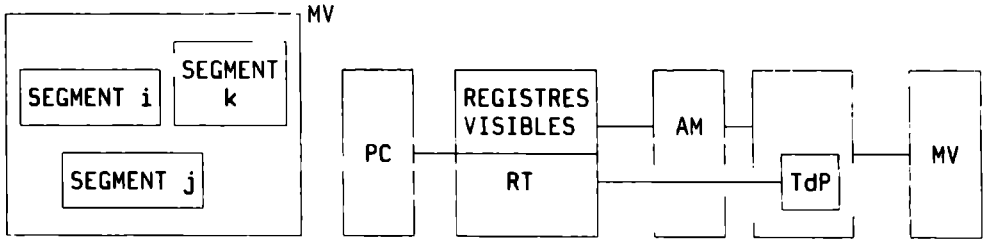
chapitre 12

gestion de la mémoire virtuelle

Le terme de virtuel se comprend mieux si on le confronte avec deux autres qualificatifs, réel et transparent:

- **virtuel**: qui est à l'état de simple possibilité selon le dictionnaire, quelque chose que l'on "voit" mais qui n'est pas là sous la forme vue, dans le langage informatique;
- **réel**: quelque chose que l'on voit et qui est bien là;
- **transparent**: quelque chose que l'on ne voit pas mais qui est là: antémémoire, registre de traduction par exemple.

La mémoire virtuelle est la mémoire que voit le programmeur, mais cette mémoire n'est plus la mémoire centrale. Elle peut être vue par le programmeur comme un espace linéaire continu (cas des systèmes à pagination seule) ou comme des ensembles d'objets indépendants appelés **segments**. Soulignons que la mémoire virtuelle est généralement plus grande que la mémoire centrale, ce point a été discuté dans le tome I et dans le chapitre 11 lors de la présentation des types de registres de traduction (RT).



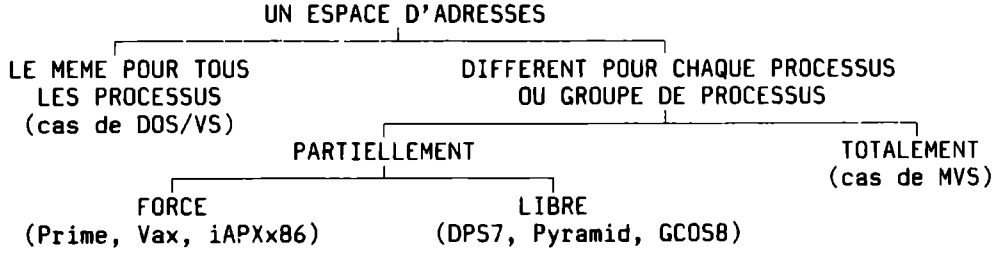
On trouve dans [WANBO]: "La mémoire virtuelle: littéralement cette phrase signifie que la mémoire principale est virtuellement infinie au point de vue taille. Fonctionnellement, elle signifie que le SE utilise l'espace disque comme une extension de la mémoire (principale) interne.". Le terme d'extension est discutable car **toute** la mémoire vue par le programmeur réside en réalité sur disque, et ce dernier ne constitue donc pas une extension de la MC; il est **la**

mémoire virtuelle, ou du moins son support. La mémoire centrale ne contient qu'un sous-ensemble du contenu du disque (ou des disques) en question, jouant en quelque sorte un rôle d'antémémoire entre le disque et le processeur central, entre la mémoire virtuelle et le processeur central. On peut cependant justifier ce terme d'extension par le fait que le disque contient, entre autres, les informations qui ne se trouvent pas en mémoire centrale, appelée aussi mémoire réelle.

Pour [BOU78] "Dans les ordinateurs utilisant l'adressage virtuel, les programmes sont écrits pour être exécutés dans une mémoire «idéale» dite MV. Les adresses repérant les informations dans cette mémoire (adresses virtuelles) sont transformées au moment de l'exécution en adresses se référant à la MC normale (adresses réelles) par un dispositif de traduction d'adresses. Les avantages obtenus par cette technique sont une plus grande souplesse de gestion de la MC en multiprogrammation, et une liberté plus grande dans l'écriture des programmes.".

12.1 – Espace d'adresses.

Un espace d'adresses (AS) est l'ensemble des zones qu'un processus peut adresser en mémoire virtuelle. Cet ensemble peut être identique pour tous les processus (cas de MVS pour les processus d'une même étape) ou différent pour chacun des processus, avec ou sans partie commune. Dans le premier cas on aura un seul espace d'adresses, plusieurs dans le second. Les processus adressent généralement une partie commune à tous les processus et une partie qui leur est privée (cas de VMS avec la région "système", de Primos, etc.):



On trouve dans [DEC83]: "L'espace des adresses virtuelles est l'ensemble des adresses vues par un programme utilisateur ... L'espace des adresses physiques est une série contiguë d'emplacements matériels adressés au niveau mot et de registres d'appareils périphériques.".

Pour accroître la protection, augmenter l'espace adressable et faciliter la communication entre espace d'adresses, donc entre processus, les 308X d'IBM ont connu une évolution d'architecture externe par rapport à leurs prédécesseurs 303X, évolution connue sous le nom de DAS (Dual Address Space). Une de ses conséquences est d'offrir une traduction d'adresses différente selon que le code ou les opérandes sont concernés, d'offrir une sorte de segmentation donc ([DUB86]). Elle correspond à:

- une extension du mot d'état (PSW) qui indique si le processus est dans son espace d'adresses "primaire" ou bien dans son espace "secondaire";
- une extension du mécanisme des clés de protection (voir chapitre 14) appelée PKM comme PSW Key Mask et liée à l'instruction PC (le PKM de l'appelant est vérifié);
- de nouvelles instructions privilégiées:
  * d'appel (PC) ou de branchement (PT) entre espaces d'adresses, analogues à ce qui se passe dans la commutation de processus sur l'iAPX286 par le biais d'instructions d'appel (CALL) et de branchement (JMP). PC est une sorte d'extension de l'instruction SVC ([DUB86]) qui permet d'appeler le superviseur; PC permet d'activer du code dans d'autres AS, d'étendre le superviseur de façon infinie;
  * de mouvement d'information vers les espaces primaires (MVCP) ou secondaires (MVCS), avec clé (MVCK):

12.2 – Segmentation.

La mémoire virtuelle est constituée d'objets appelés segments. Un segment est un ensemble d'informations de même **nature**: segment de code avec des instructions, segment de constantes avec des données non modifiables, segment contenant une structure, un tableau, ou même un fichier sur Multics. Le segment est encore un ensemble d'informations avec les mêmes **protections** d'accès: selon qu'il fait ou non partie de l'espace adressable du processus, en fonction d'autres restrictions internes à l'espace du processus.

Le segment est donc l'élément de gestion de la mémoire virtuelle, tout comme la page est l'unité de gestion de la mémoire centrale. Contrairement à la page, le segment est de **taille variable**, puisque liée à celle de l'information qu'il contient. Cette taille peut être exprimée en octets ou en multiples de 2^n octets selon l'architecture de l'ordinateur. La figure 12c montre que l'iAPX386 offre deux "granularités" pour définir la taille des segments, par octet ou par page.

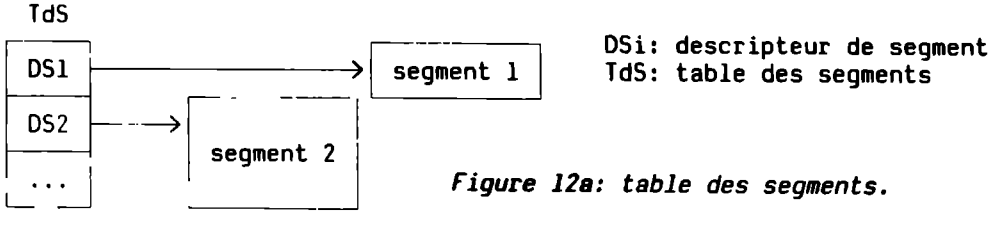


Figure 12a: table des segments.

Tout comme les tables de page TdP dans la pagination, la segmentation fait apparaître des **tables de segment** TdS avec une entrée par segment, entrée fréquemment appelée **descripteur**, avec le sens défini dans le tome I. L'ensemble des descripteurs définit l'espace adressable par le processus qui en dispose (figure 12a).

On trouve dans [CRO75] (page 75): "Un segment est un objet constitué d'une suite linéaire d'emplacements numérotés 0, 1, 2, etc. Il sert à regrouper des objets de même nature, de même protection, de même durée de vie, etc. Il est généralement la plus petite unité partageable (c'est-à-dire accédée par plusieurs processus) d'un système [...] Une procédure, un fichier, un tableau, une pile sont des exemples d'objets rangés dans un segment.". Notons que la caractéristique de "plus petite unité partageable" n'est pas toujours vraie, comme dans les DPS7 ou l'iAPX432.

Sur l'iAPX432 chaque module de programme translatable de façon indépendante reçoit au moment de son exécution (run time) les numéros des segments auxquels il a le droit d'accéder (l'environnement d'accès du module). L'iAPX432 emploie le terme de "mémoire structurée" parce que Intel estime que son mécanisme de segmentation est différent de la simple segmentation, une des différences étant le segment des accès (des capacités) sur lequel nous reviendrons dans le chapitre 14.

Pour donner quelques exemples de taille des espaces adressables, nous donnerons d'abord la taille de l'élément de base de l'espace adressable, le segment. La taille maximum de ces segments est de:
- 64 Ko ("petits" segments) ou 4 Mo ("grands" segments) sur DPS7, 64 Ko ou 16 Mo sur Z80000,
- 64 Ko sur iAPX432, iAPX286,
- 128 Ko sur PRIME,
- 256 Ko sur ICL 2900,
- 512 Ko sur MV/X000 de Data General,
- 1 Kmot de 48 bits sur Burroughs 5500,
- 256 Kmots de 36 bits sur Multics,
- 64 Mo sur DPS8 avec GCOS8,
- 4 Go sur l'iAPX386.

Chaque **processus** peut adresser:
- 128 Ko sur Sintran III,
- 512 Ko sur Tandem16 (256 pour le système, 256 pour l'utilisateur),
- 1 Mo sur PDP10 avec pagination,
- 16 Mo sur 370 avec MVS, 3B20 d'AT&T, 68010 de Motorola avec MMU,
- 16 ou 256 Mo selon les modèles sur Apollo,
- 252 Mo sur DPS7,
- 32 Mo privés sur PRIME par utilisateur, 512 Mo globalement,
- 1 Go sur iAPX286 en mode virtuel,
- 2 Go sur 370 avec MVS/XA, sur VAX, sur MV/X000 (4 Go adressables, mais 2 Go laissés pour l'utilisateur), sur Z80000 en mode segmenté,
- 4 Go par module de programme sur iAPX432, sur 90X de Pyramid, sur GEC 63/30 ou 40, sur ICL 2900,
- 8 Go sur DPS8 avec mémoire virtuelle (GCOS8), avec NOS/VSE,
- 256 To sur l'iAPX386 (adressage avec 48 bits, 2^{16} segments par processus), sur S/38.

12.3 – Support matériel.

Afin d'accélérer la consultation des tables de segment, on utilise des registres de traduction RT analogues à ceux que nous avons présentés dans le cadre de la pagination. Nous les illustrerons par quelques exemples.

Dans le tome I nous avions laissé le 8086 avec ses quatre registres de base CS, DS, SS et ES, visibles de l'utilisateur au niveau de l'architecture externe, non sans avoir souligné qu'il ne s'agissait pas d'une segmentation authentique. L'iAPX286 comporte toujours ces quatre registres de 16 bits puisqu'il est "compatible vers le haut" avec le 8086, mais ils n'y sont plus que des sélecteurs de segment: ce ne sont plus eux en effet qui contiennent les adresses où sont chargés les segments en mémoire centrale.

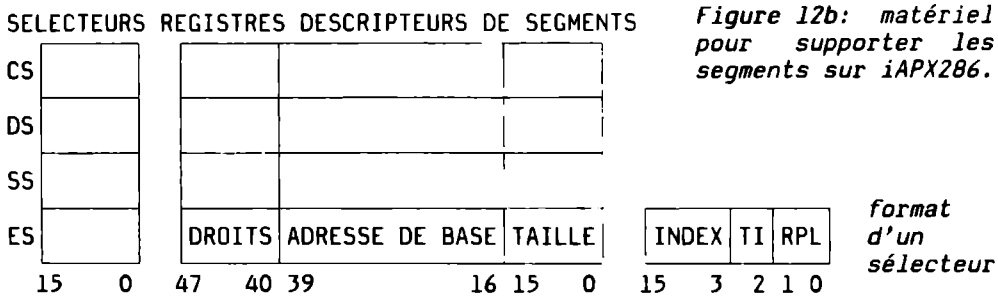


Figure 12b: matériel pour supporter les segments sur iAPX286.

TI : Table Indicator, GDT (0), LDT (1).
RPL: Requested Privilege Level (voir chapitre 14).

Ils ne contiennent qu'un numéro de segment appelé "index" dans la figure 12b, d'où 2^{14} segments par processus dans l'iAPX286. L'adresse de base, les droits d'accès, la taille réelle, sont rangés dans des registres descripteurs invisibles de l'interface instruction et qu'Intel appelle antémémoire: ils forment en effet une antémémoire de 4 descripteurs, antémémoire par rapport aux tables de segments. Ils jouent le rôle des registres de traduction RT. Dès qu'un numéro de segment est chargé dans un des registres sélecteurs, le registre descripteur associé est automatiquement rempli par le matériel. On voit ainsi apparaître une vraie segmentation sur l'iAPX286 avec les champs "droits" et "taille" qui n'existent pas sur le 8086. Il y a en fait 8 et non pas 4 registres dans cette antémémoire afin d'accéder en plus aux segments qui contiennent les tables de segments (LDT et GDT, le bit TI départageant les deux tables), la table des interruptions IDT, le contexte du processus TSS.

L'**iAPX386** comporte 6 registres sélecteurs au lieu des 4 de l'iAPX286, les descripteurs y ont 64 bits au lieu de 48. L'unité de gestion de la mémoire (MMU) intégrée sur la même puce, contient entre autres:

- une "unité de segmentation" contenant ces descripteurs de segment, chargée de vérifier les droits d'accès et de transformer l'adresse virtuelle en adresse linéaire;

- une "unité de pagination" contenant les 32 registres de traduction associés à la pagination, chargée de transformer l'adresse linéaire fournie par l'unité de segmentation en adresse physique. Cette unité de pagination fait appel à une technique de pagination à deux niveaux illustrée dans le chapitre 11. Les 10 bits du champ "hyperpage" dans l'adresse linéaire permettent de référencer une des 1024 entrées de la table des hyperpages qui occupe 4 Ko, soit une page. Chacune des tables de pages comprend de même 1024 entrées de 4 octets et occupe donc aussi une page.

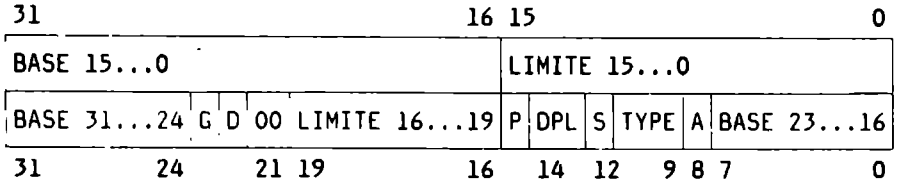
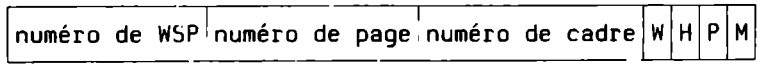


Figure12c: structure réelle d'un descripteur de segment sur l'iAPX386 (8 octets).

BASE : adresse de base du segment
LIMITE: taille du segment
P : segment présent (1) ou non
DPL : anneau d'accès (0 à 3)
S : type de segment, système (0) code ou donnée (1)

TYPE: sous-type de segment
A : référencé ou non
G : granularité, page (1) ou octet (0)
D : code 32 bits (1) ou 16 bits (0)

Les DPS8 sous GCOS8 ont 64 registres de 45 bits par processeur, rangés en 4 lignes de 16 registres. Ces RT peuvent être ignorés (bypassed) au niveau du matériel, par un commutateur, ou bien au niveau du logiciel. La segmentation y est matérialisée par des registres descripteurs de segment.



RT du DPS8.

12.4 – Gestion MC avec segmentation seule.

Quand la pagination n'est pas utilisée, la MC contient des segments complets et chacun est stocké de façon contiguë en mémoire. On retombe sur les problèmes évoqués plus haut pour les partitions translatables. Elle présente les **avantages** suivants pour la gestion de la mémoire centrale:

- la fragmentation MC externe est éliminée car on peut tasser les segments, il y a peu ou pas (granularité d'un octet sur iAPX386 par exemple) de fragmentation interne;
- l'espace adressable peut dépasser la taille de la MC;
- la taille des segments peut croître de façon dynamique, dans les deux directions sur l'iAPX386;

- on peut détecter les débordements: le fait de générer un déplacement supérieur à la taille courante du segment provoque une interruption;
- le chargement dynamique des segments (à la demande) est réalisable;
- le partage du code et des données est facilité;
- l'accès au segment est contrôlable.

Pour gérer la mémoire centrale la segmentation seule a comme **défauts**:
- la taille courante d'un segment ne peut dépasser la taille de la MC;
- la taille d'une structure (tableau, procédure) est limitée par la taille maximale du segment. Ainsi sur PRIME, le segment est de 128 Ko et les procédures ne peuvent dépasser 128 Ko; pour les tableaux il est par contre possible de dépasser cette limite. Sur DPS7, les segments ont soit 64 Ko (petits segments), soit 2 Mo (grands segments); les tableaux ne peuvent pas dépasser ces quantités. Dans certains processeurs il est cependant possible de faire des sauts (jump) d'un segment à un autre (iAPX d'Intel);
- le tassement (compaction) en MC représente une dépense de temps processeur;
- il est plus difficile de gérer des éléments de taille variable sur mémoires centrale et secondaire;
- la traduction d'adresse et les exceptions dues aux "fautes" de segment prennent du temps en cas de segmentation dynamique;
- il y a un risque de dérapage (emballement, segment thrashing) dans les gestions dynamiques.

SEGMENTATION	SEGMENTATION + PAGINATION
BURROUGHS 5000	MULTICS
BULL 64	PRIME, ICL2900
DPS7	DPS8 sous GCOS8
	MV/X000 de Data General (4)
MC68000	iAPX 386 (1)
Z8000	Z80000 (2)
iAPX 286	WE32101 (3)
	ICL 2900 (5)
	NOS/VE

Tableau 12d: exemples de systèmes avec segmentation.

(1) Segmentation et pagination (à deux niveaux) peuvent être activées ou non, d'où quatre cas de fonctionnement.
(2) La segmentation peut être désactivée, elle peut utiliser un ou deux niveaux de tables de segments.
(3) Segmentation et/ou pagination.
(4) Forme dégradée, un segment y est en fait un anneau.
(5) La pagination peut être désactivée segment par segment.

12.5 – Protection.

La segmentation offre naturellement un mécanisme de protection. Dans les tables de segments (TdS), on trouvera des entrées avec la structure suivante:

P	taille	adresse

où P représente les protections du segment contre tout accès non autorisé, tout comme dans le cas des tables de page.

Dans l'iAPX432 les droits d'accès sont stockés dans les tables d'accès de chaque module de programme. L'iAPX286 distingue quatre types de segment: exécutable seulement, exécutable et lisible, lisible seulement, lisible et modifiable (read and write); il faut donc deux bits pour coder ces quatre possibilités. Un segment de code (CS) sera ainsi défini comme exécutable et le chargement vérifiera cette caractéristique. Un segment pile (SS) pourra lui être lu et écrit. Le chapitre 14 détaillera ces protections.

12.6 – Segmentation plus pagination.

Si la segmentation convient à la gestion de la mémoire virtuelle et la pagination à celle de la mémoire centrale, il est logique de les combiner pour bien gérer ces deux objets. Un nombre croissant de machines implémente ces deux mécanismes comme le montre le tableau 12d.

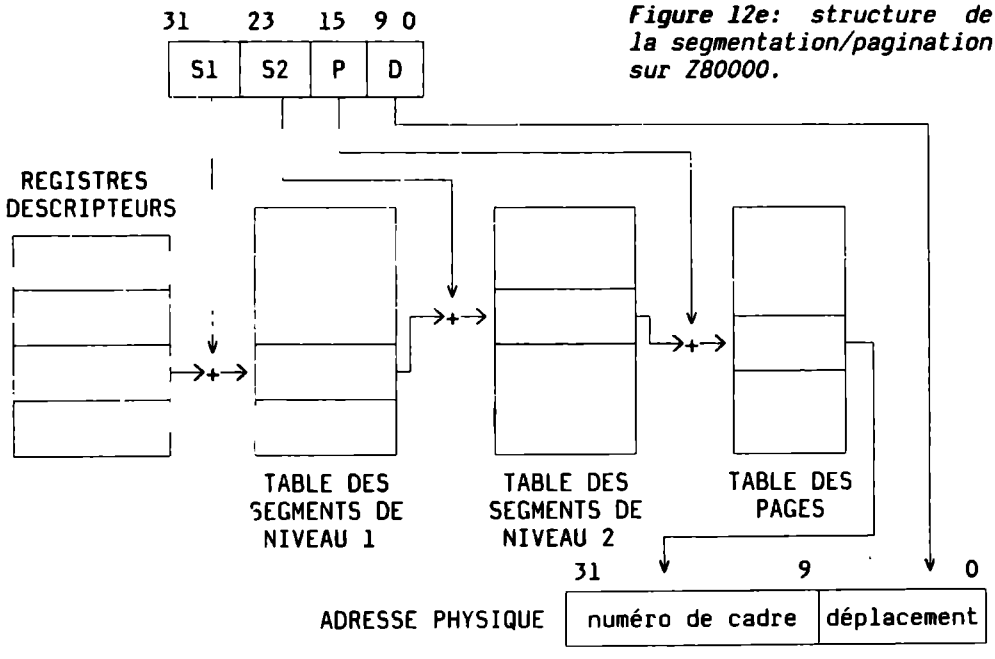


Figure 12e: structure de la segmentation/pagination sur Z80000.

12.7 – Mémoire à un seul niveau.

Segments et fichiers sont des objets que seule la technique d'accès différencie: les informations contenues dans les segments sont localisées par les modes d'adressage, les segments l'étant par le mécanisme de segmentation; les données dans un fichier sont localisées

par les méthodes d'accès, les fichiers par le SGF. Rien n'empêche de fusionner ces deux objets en créant un mécanisme d'accès unique. C'est ce que l'on trouve dans Multics, le S/38 d'IBM (adressage sur 48 bits, d'où $256*10^{12}$ objets possibles). L'orientation objet du S/38 (chapitre 10) qui peut être considérée comme une généralisation de la segmentation, fait que l'utilisateur ne voit qu'un seul espace d'objets. Le dispositif "Data in Virtual" annoncé dans MVS/XA 2.2 va dans le même sens puisqu'il permet de placer (map) un fichier dans l'espace des adresses virtuelles, les routines d'ASM remplaçant les macros READ/WRITE ainsi que la gestion des tampons.

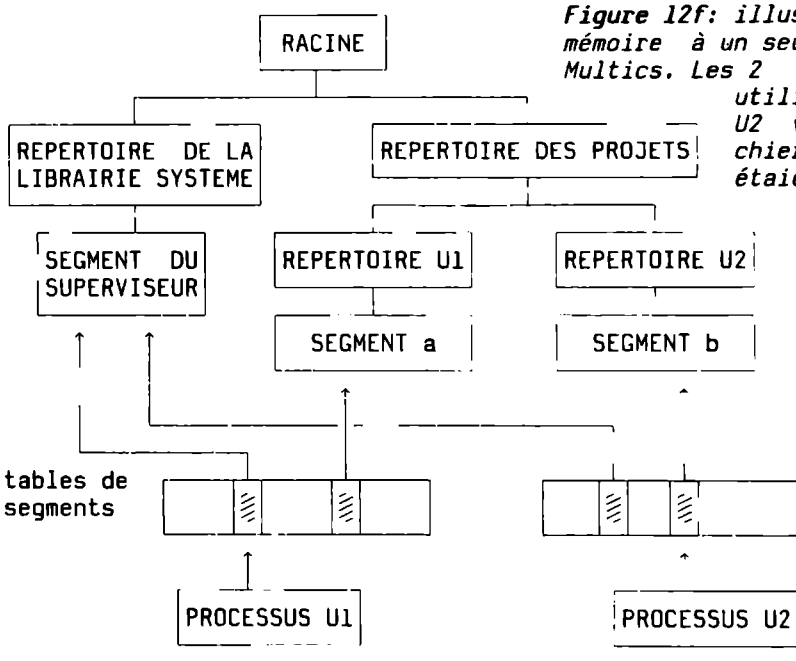


Figure 12f: illustration de la mémoire à un seul niveau de Multics. Les 2 processus des utilisateurs U1 et U2 voient les fichiers comme s'ils étaient en mémoire virtuelle.

12.8 – Mémoire virtuelle distribuée.

La mémoire virtuelle adressable par un processus est distribuée sur un ensemble de machines interconnectées. Domain d'Apollo en est un exemple avec un espace adressable de 2^{96} octets, accessibles par des stations de travail monopostes DNx00 reliées par un réseau local en boucle. 64 des 96 bits de l'adresse servent à coder le nom interne (UID) d'un objet de 2^{32} octets, nom qui est unique dans le temps et dans l'espace. Une fois localisé par le serveur de noms, l'objet est ensuite accédé page par page, en fonction de la demande:

commande de l'utilisateur ⟶ [SERVEUR DE NOMS] ⟶ UID

12.9 – Exemples de structure de MV.

La première machine à mémoire virtuelle par pagination fut l'ATLAS. C'est la première qui distingua:
- l'espace adressable par programme,
- l'espace des adresses physiques (mémoire centrale physiquement configurée sur le système).

12.9.1 – Cas des DPS7.

La figure 12g illustre ces tables de segment dans le cas du DPS7 de Bull. Le registre BAR (Base Address Register) contient la "racine" de toutes les structures du système puisqu'il donne l'adresse d'une table permettant d'accéder aux processus, aux segments généraux, aux vecteurs d'exception (interruption), etc.

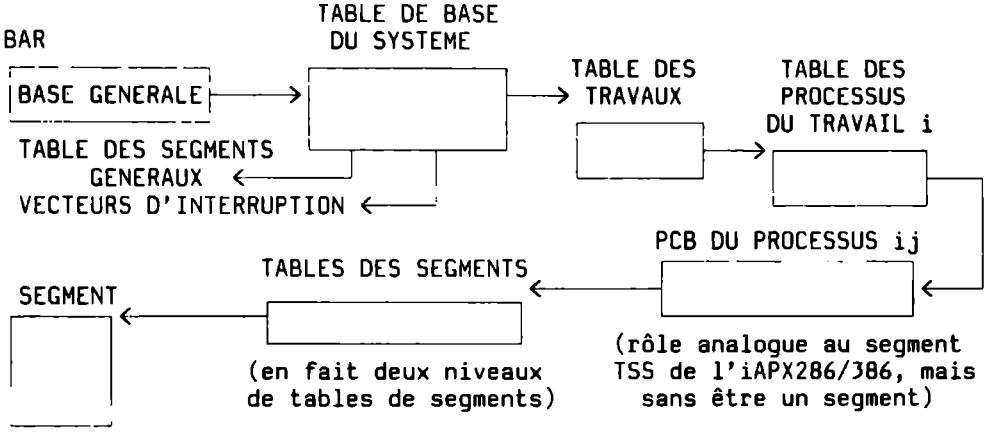


Figure 12g: structures du DPS7 liées à la segmentation.

On notera au passage qu'aucune zone mémoire imposée à l'avance n'est requise, ce qui est une source de meilleure résistance aux incidents, pannes de mémoire en l'occurrence. Les groupes de processus y sont appelés travaux (job), et une table contient la liste des processus actifs. Le SE apparaît lui-même sous forme de tels travaux. Chaque travail dispose d'une table recensant ses processus actifs. Intéressons-nous au processus d'un travail. La table des processus à laquelle il appartient contient un pointeur vers son contexte appelé PCB (Process Control Block). Ce PCB contient deux pointeurs vers deux tables, l'une associée aux petits segments (64 Ko), l'autre associée aux grands segments (2 Mo).

12.9.2 – DOS/VS.

Les partitions du DOS qui résidaient en mémoire centrale se retrouvent en mémoire virtuelle dans DOS/VS, elles sont appelées BG-V, Fi-V comme BackGround et Foreground dans la figure 12h. La SVA (Shared Virtual storage Area) est utilisée pour stocker:

- les méthodes d'accès,
- des listes de pointeurs vers la bibliothèque des programmes exécutables (CIL pour Core Image Library).

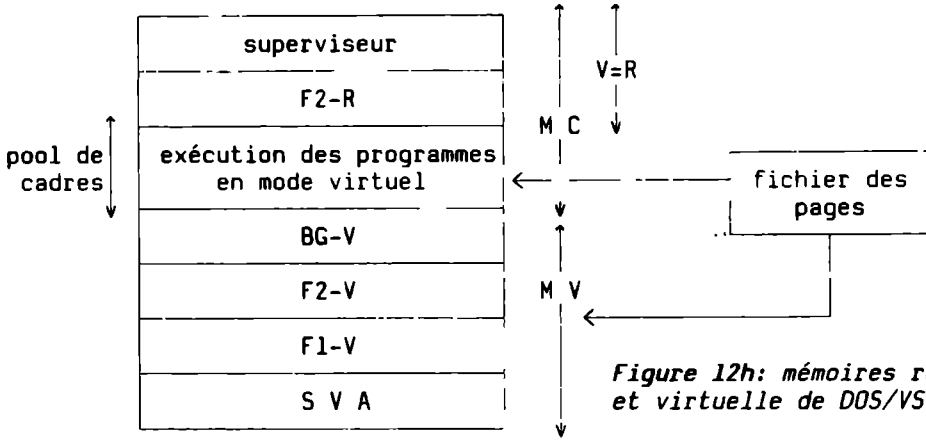


Figure 12h: mémoires réelle et virtuelle de DOS/VS.

12.9.3 – MVS.

L'espace adressable est de 16 Mo sur les machines à adressage 24 bits (256 hyperpages de 16 pages de 4 Ko) et 2 Go sur les machines avec adressage de 31 bits. L'espace disponible pour chaque utilisateur est celui laissé libre par le système. Sur les machines à adressage 24 bits, l'espace disponible pour l'utilisateur arrivait à ne pas être suffisant, en particulier dans les systèmes transactionnels comme IMS. Le registre CR1 (ou STOR, Segment Table ORigin) donne l'adresse de la table des hyperpages.

Un des rôles des SRB (System Request Block) sous MVS est de résoudre le problème d'adressabilité, de communication, entre ces espaces d'adresses.

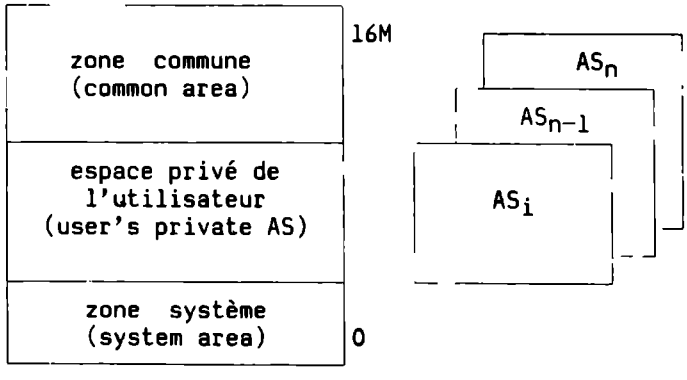
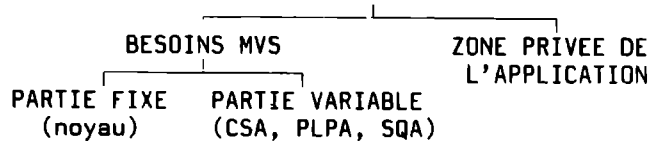


Figure 12i: carte de la MV sous MVS.

12.9.3.1 – Adressage 24 bits (MVS/370).

Chaque espace d'adresses comprend 16 Mo partagés en deux parties: l'une pour les besoins de MVS, l'autre constituant la zone privée de l'utilisateur. On retrouve ainsi un PCP (chapitre 11) dans chaque AS. La zone MVS comprend plusieurs parties dont la taille est liée à la configuration (le noyau) ou au volume et à la nature de la charge supportée (CSA, PLPA et SQA dans l'ordre selon [DUB86]).



Structure d'un espace d'adresses sous MVS.

12.9.3.1.1 – Zone commune.

La zone commune, partagée par tous les utilisateurs (tous les AS), est allouée depuis les adresses hautes de la mémoire. Elle a la structure ci-dessous:

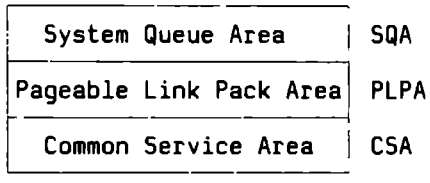


Figure 12j: structure de la zone commune sous MVS.

- la **SQA**, zone de travail (en clé zéro), contient des tables (table des hyperpages, table des pages de la zone système et de la zone commune, table APF par exemple) et des listes (queues) relatives au système tout entier comme les ASCB, PCB (Page Control Block, 64 octets pour toute page en cours d'échange entre MC et MV). Ses cadres sont fixés à long terme, ce qui signifie dans la terminologie MVS que les pages ne sont **pas paginables** ni **vidables** ("non-swappable", on ne peut pas les vider sur mémoire secondaire).
- la **PLPA**, chargée depuis le fichier SYS1.LPALIB, contient des programmes du système, réentrants comme les SVC, les méthodes d'accès (AM), ainsi que les pointeurs pour accéder à ces modules (LPDE, Link Pack Directory Entry). Elle peut contenir des programmes réentrants des utilisateurs (cas de commandes TSO). Elle est initialisée par un RIM (chapitre 16) lors du démarrage du système, sa taille ne varie plus ensuite. Cette zone est **paginable** mais **résidente** (pas de "physical page-out" puisque le code est réentrant et n'est donc pas modifié);
- la **CSA** contient des données des utilisateurs privilégiés. Les données associées à un espace d'adresses donné peuvent y être protégées par les clés de protection qui ne sont pas en nombre suffisants: la clé 8 est partagée par tous les zones en V=V, les clés 9 à 15 sont utilisées par les zones en V=R. **Paginable**, elle est utilisée pour

communiquer entre espaces d'adresses privés privilégiés par la méthode "standard" de MVS (celle des SRB, cf. chapitre 9): à un moment donné un seul espace est adressable, celui dont l'adresse de la table des hyperpages est dans le registre CR1. TCAM, JES2, JES3, VTAM, le superviseur utilisent la CSA dans ce but. [DUB82] souligne que cette communication à travers la CSA est "déficiente" jusqu'à MVS/SP1.2, surtout à cause de l'inflation mémoire ([DUB6]).

La zone commune peut aussi contenir des sous-zones optionnelles:

- la **MLPA** (Modified LPA) qui comprend des modules réentrants des librairies système ou utilisateurs. Ces modules ne sont pas obligatoirement "read only" (lisibles seulement), mais peuvent être modifiées, dans un but de mise au point par exemple: on peut ainsi y tester des modules avant de les mettre dans la PLPA. La MLPA constitue une extension de la PLPA mais n'est pas sauvegardée entre deux démarrages du système (IPL);
- la table **BLDL** paginable (pageable BLDL table): elle contient une liste dans laquelle sont rangées des entrées de répertoires pointant vers des modules de librairies système comme SYS1.LINKLIB, afin d'augmenter les performances, puisqu'elle constitue un morceau de répertoire en mémoire centrale et non plus sur disque. On y met par exemple les références de modules très utilisés mais non réentrants ou trop gros pour être rangés en FLPA ou PLPA. [FRI81] montre qu'avec une BLDL, les déplacements des bras (seek, voir tome IV) passent de 33 à 14 millisecondes sur un 3350 pour le chargement des programmes (load module); d'autres gains sur le délai rotationnel et les temps d'attente font globalement passer le chargement (FETCH) de 179 à 106 ms.

MVS/SP1.3 utilise la fonction "cross memory" (ou DASF) pour permettre des transferts directs de données entre les zones privées de deux AS, ainsi que l'exécution directe de programmes dans la zone privée d'un autre AS. L'utilisation des zones CSA et surtout PLPA peut ainsi être réduite et la protection des données est améliorée puisqu'il est plus difficile d'atteindre une zone privée que la CSA.

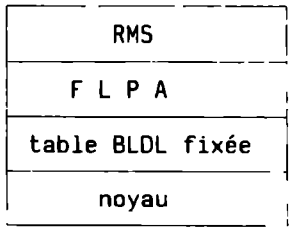


Figure 12k: structure de la zone système de MVS avec les sous-zones optionnelles.

12.9.3.1.2 – Zone système.

La zone système est définie à l'initialisation du système (IPL) et contient le **noyau** (nucleus) du système (SCP System Control Program): elle en constitue la partie résidente et sa taille reste fixe. Elle fait partie de chaque espace d'adresses (AS) et y prend 6 à 8 Mo dans MVS "release" 3.8 (mai 80, elle dépend en fait de la

configuration); dans SVS le noyau partageait déjà l'espace adressable avec tous les programmes des utilisateurs, mais il n'y avait qu'un espace d'adresses. Cette zone est allouée depuis les adresses basses de la mémoire et comprend:
- les programmes de traitement des interruptions (interrupt handler),
- le distributeur (dispatcher), gestionnaire des tâches,
- le GESP (IOS),
- le mécanisme de pagination,
- les SVC de type 1.

Elle n'est ni paginable, ni vidable ("non swappable"); on dit que c'est une zone V=R (adresse virtuelle égale à l'adresse réelle). En plus du noyau, cette zone système contient:
- la table des cadres (page frame table entries) utilisée par RSM,
- les DEB (Data Extent Block) pour les librairies du système,
- les blocs de contrôle d'ASM,
- les UCB qui correspondent à la description de la configuration, avec un UCB par unité ou chemin y conduisant,
- les modules de reprise (recovery).

Elle peut encore contenir des zones optionnelles chargées depuis SYS1.SVCLIB ou SYS1.LINKLIB:
- la FLPA (Fixed Link Pack Area), extension géographique de la PLPA, pour des modules réentrants ou en lecture seulement. Ces modules sont fixés (V=R, PGFIXed) et on y met généralement des modules très utilisés ou des modules qui doivent être très rapidement exécutés pour assurer de bons temps de réponse, pour éviter de perdre des cycles processeur à fixer (PGFIX) ou libérer (PGFREE) les pages (cas de STIMER);
- une table des BLDL, fixée en mémoire, qui joue un rôle analogue à la table de même nom mise en zone commune;
- RMS, une extension du noyau, qui contient des routines pour améliorer la disponibilité offerte par MVS.

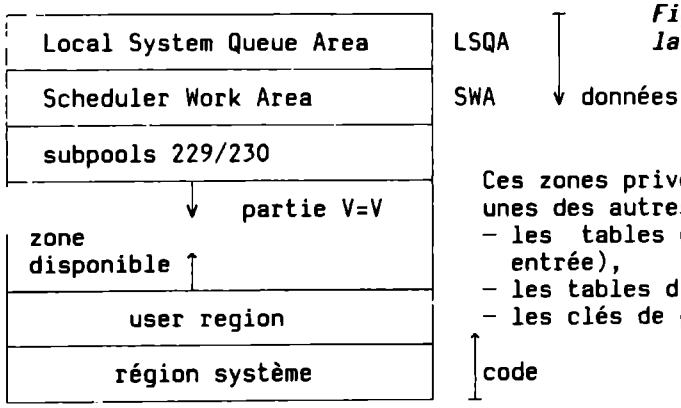


Figure 121: structure de la zone privée sous MVS.

Ces zones privées sont protégées les unes des autres par:
- les tables de page (2 octets par entrée),
- les tables d'hyperpages,
- les clés de protection.

12.9.3.1.3 – La zone privée.

La zone privée (de 8 à 10 Mo en moyenne, toujours plus de 5 Mo,

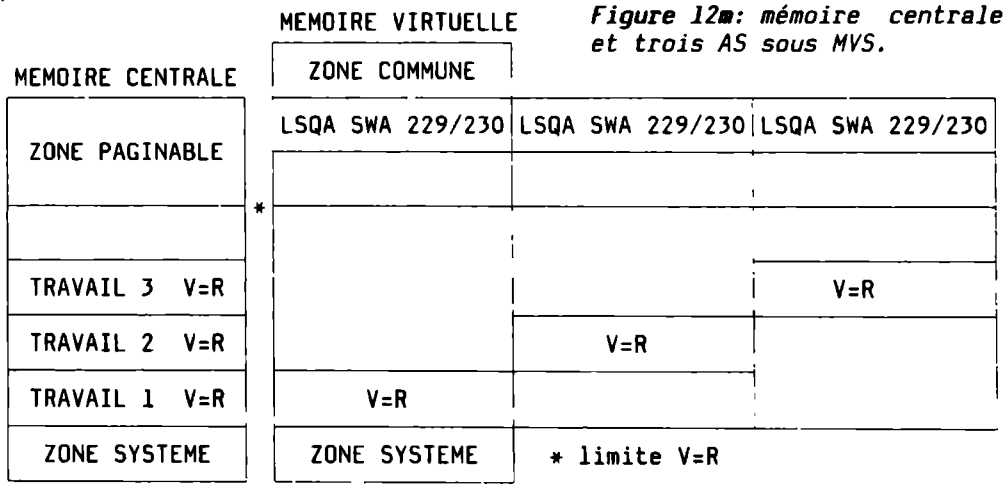
variable selon les sites) réservée à l'utilisateur a la structure donnée figure 121.

La **LSQA**, zone de travail locale à un AS, construite par VSM lors de la construction de l'AS, est fixée en mémoire centrale lorsque l'AS n'est pas vidé (on dira lorsque l'AS est "swapped in"). La LSQA est apparue avec TSO sous MVT. Elle contient les blocs de contrôle requis par la RCT (Region Control Task) qui est la tâche de plus haut niveau de l'AS (Address Space). La RCT joue un rôle clé en cas de vidage (swapping). La LSQA est paginable et vidable (swappable).

La **SWA** contient des blocs de contrôle qui existent depuis l'initialisation de la tâche jusqu'à sa terminaison, blocs qui ne concernent qu'un seul travail (job): blocs de contrôle de fichier, d'allocation de périphérique. Elle contient des informations autrefois stockées dans le fichier des travaux lus (SYS1.SYSJOBQE); ce fichier était une source de ralentissement dans les anciens SE MVT et SVS. La SWA joue un rôle de liste locale des travaux (local job queue) pour chaque AS en contenant les cartes JCL du travail en cours d'exécution.

Un **sous-pool** (subpool) est un groupe logique de blocs MC qui ont des caractéristiques communes ainsi qu'un numéro qui l'identifie. Les sous-pools 229 et 230 contiennent des blocs de contrôle comme les DEB. Le 229 est protégé pour le chargement (bit Fetch dans la clé de protection).

Figure 12m: mémoire centrale et trois AS sous MVS.



La **région utilisateur** (user region) correspond à une zone virtuelle (V=V) paginable et vidable. Elle contient les pages qui ont été chargées en MC (paged-in). Sa taille peut être limitée par le paramètre REGION des commandes // JOB ou // EXEC.

Une zone réelle (V=R) se situe juste après le noyau pour chaque travail ayant une portion non paginable, les adresses virtuelles et réelles y sont identiques. Sa taille est aussi définissable par le paramètre REGION. Ces zones V=R son protégées par les clés 9 à 15.

La **région système** contient des fonctions du système qui travail-

lent pour l'AS et s'exécutent sous la tâche RCT (Region Control Task). Elle a un rôle analogue à celui de la zone SSA de GCOS3.

La figure 12m illustre le rapport entre mémoire réelle (MC) et les différents espaces d'adresses de la mémoire virtuelle. La figure 12n montre la distance qui sépare MVS des OS de la série 360.

VSM (Virtual System Manager) contrôle le découpage de la mémoire virtuelle de chaque espace d'adresses. ASM (Auxiliary Storage Manager) gère les fichiers contenant les pages; ASM gère aussi les fichiers de vidage (swap files) pour les espaces d'adresses vidés.

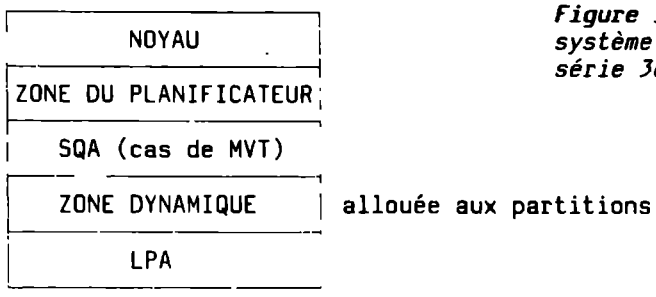


Figure 12n: carte mémoire d'un système d'exploitation OS de la série 360.

12.9.3.2 – Adressage étendu (31 bits, MVS/XA).

Dans les premiers systèmes MVS, au milieu des années 70, la zone MVS n'atteignait pas les 8 Mo restants, les applications étaient loin de nécessiter 8 Mo et les 16 Mo de l'espace d'adresses paraissaient luxueux (cf. [SMI82]). Mais MVS a grossi, les applications aussi, particulièrement dans les environnements transactionnels et une extension est devenue indispensable. La CSA en particulier a grossi, du fait de TSO qui a tendance à y ranger des commandes pour améliorer les performances, du fait des méthodes d'accès modernes comme VTAM qui constitue un AS et échange les messages avec les autres AS en passant par la CSA. L'adressage 31 bits correspond à l'**adressage étendu** (XA comme Extended Architecture) dans la terminologie IBM.

	MEMOIRE VIRTUELLE		MEMOIRE CENTRALE	
	XA	370	XA	370
PSA	4	4		
FLPA	0	28		
FBLDL	8	8		
NOYAU	328	1096	1584	1136 (1)
MLPA	572	608		
PLPA	3574	4290	2228	2164
CSA	2033	2029	2292	2340
SQA	383	615	1944	1376
HSA			376	448

Tableau 12o:exemples de taille mémoire (en Ko) selon [BAU83] pour une charge de 270 AS TSO et 15 AS en lot.

(1) avec PSA, FLPA, FBLDL
HSA: microprogrammes des 308X en MC
MVS/SP1.3.2 en 370
MVS/SP2.1.0 en XA

Le noyau n'est plus dans une zone V=R mais dans une zone V=F (comme Fixe): les adresses virtuelles et réelles peuvent maintenant être différentes, de plus le noyau n'occupe plus obligatoirement des cadres contigus. Il reste cependant non paginable (il n'y a pas de faute de page) mais il n'est plus nécessaire d'avoir une zone mémoire contiguë sans défaut pour stocker le noyau; le système résiste mieux aux pannes. De nombreuses machines offrent maintenant ce genre de dispositif permettant de ranger le système ou certaines de ses tables n'importe où en mémoire centrale, nous l'avions signalé pour les DPS7. Un contre exemple en est le 8086 dont la table des vecteurs d'interruption (voir tome III) doit obligatoirement commencer à l'adresse zéro.

[BAU83] donne des comparaisons entre MVS/370 (adressage 24 bits) et MVS/XA (adressage 31 bits) qui concluent à une moindre occupation de l'espace adressable en dessous des 16 Mo par le système en XA (2 Mo de moins), mais à une plus grande occupation de la mémoire centrale, ce qui est cohérent avec le fait que les machines XA ont une mémoire configurable supérieure: 2 Go contre 64 Mo sous MVS/SP1.3 (tableau 12o).

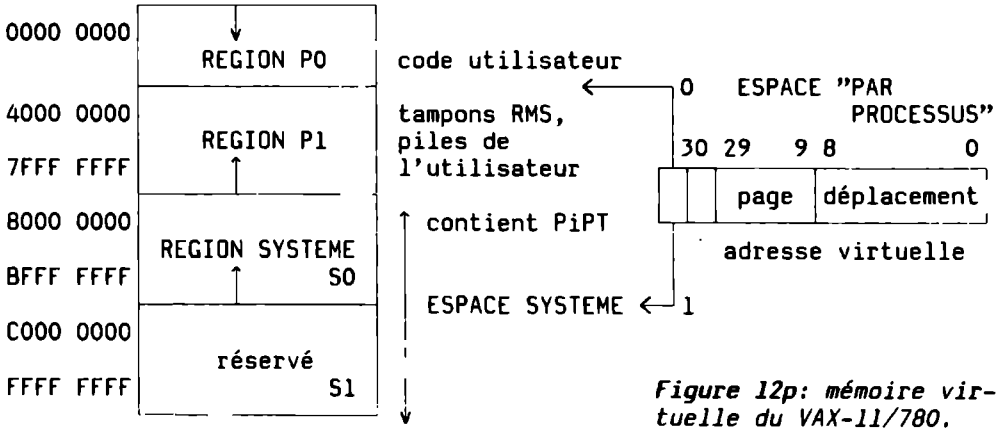


Figure 12p: mémoire virtuelle du VAX-11/780.

12.9.4 – VAX/VMS.

L'espace est décomposé de façon classique entre un espace propre à chaque processus de 1 Go et un espace commun à tous les processus, l'espace système, de 1 Go lui aussi; le bit de gauche de l'adresse virtuelle définissant lequel des deux espaces est adressé (figure 12r). Chaque table des pages, SPT pour la région système (System Page Table), P0PT et P1PT pour les régions P0 et P1, dispose de deux registres de base et longueur, [SBR,SLR], [PiBR,PiLR]; ces registres permettent au matériel de localiser les tables et de vérifier que l'accès ne se fait pas en dehors de la zone réellement utilisée, modulo 512, la taille de la page:

12.9.5 – VM chez IBM.

Sous VM il convient de distinguer trois niveaux de mémoire:
- la mémoire physique, aussi appelée mémoire réelle, mémoire centrale,
- le second niveau correspondant à la mémoire principale d'une machine virtuelle,
- le troisième niveau qui est la mémoire virtuelle créée par une machine virtuelle comme DOS ou MVS, celle vue par une application.

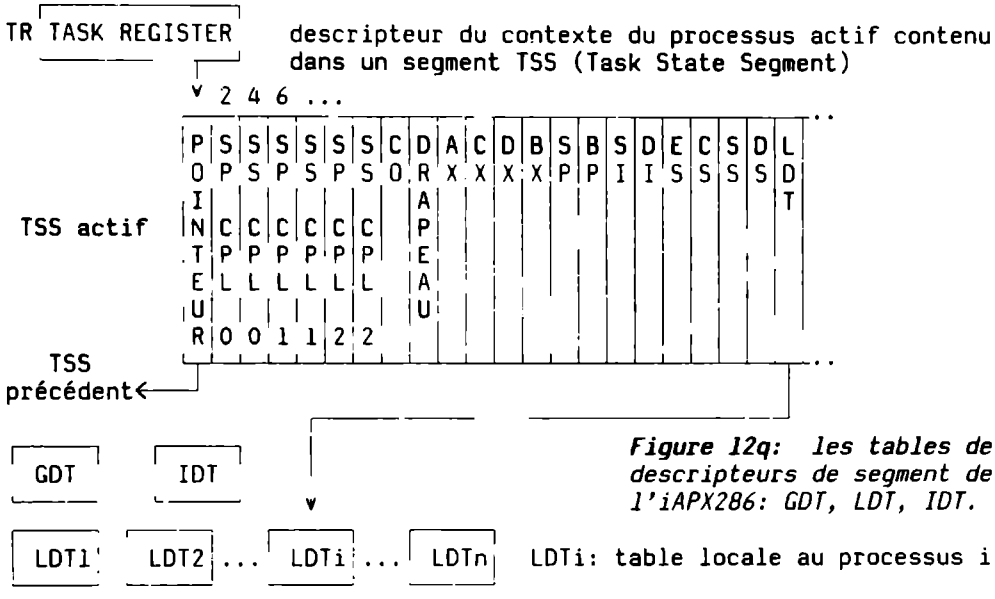


Figure 12q: les tables de descripteurs de segment de l'iAPX286: GDT, LDT, IDT.

12.9.6 – L'iAPX286.

L'espace adressable d'un processus est défini par trois types de tables de descripteurs de segment:
- une table des **descripteurs globaux** (GDT, Global Descriptor Table) correspondant à des informations communes à tous les processus, analogue à la "zone commune" de MVS, encore que l'aspect commun soit réalisé par l'adressage en MC dans MVS, par l'adressage en MV dans l'iAPX286. L'instruction LGDT (Load GDT) charge les registres base-limite définissant la GDT pour le processeur central. La GDT contient tout type de descripteur de segment, à l'exception des descripteurs d'interruption et de trappe (cf. chapitre 14);
- une table des **descripteurs d'interruption** (IDT) qui généralise les 256 vecteurs d'interruption du 8086. L'instruction LIDT (Load IDT)

charge les registres base/limite qui définissent l'IDT. Cette table est accédée par l'instruction INT ou par les interruptions externes et internes (exceptions, trappes selon Intel). Seuls des descripteurs de processus (task gate), d'interruption et de trappe y sont contenus;
- une table des **descripteurs locaux** (LDT), une par processus. LLDT permet de charger le sélecteur LDTR. Elle ne contient que des descripteurs de segment, de tâche et d'appel (call gate).

Un processus ne peut adresser un segment dont le descripteur ne figure pas dans l'une de ces 3 tables. Pour accéder à ces tables, l'iAPX286 comporte 3 registres descripteurs supplémentaires qui s'ajoutent aux 4 registres descripteurs de l'antémémoire associés à CS, DS, SS et ES (cf. figure 12b): le registre descripteur définissant la LDT et associé à un registre sélecteur LDTR, les registres descripteurs définissant la GDT et l'IDT qui n'ont pas de registre sélecteur associé puisque ces tables sont uniques. Le processus actif est défini par le sélecteur TR. On notera que par rapport au 8086 on est passé de 1 à 4 piles, une par anneau, la notion d'anneau étant représentée par CPL (Current Privilege Level) sur la figure 12q.

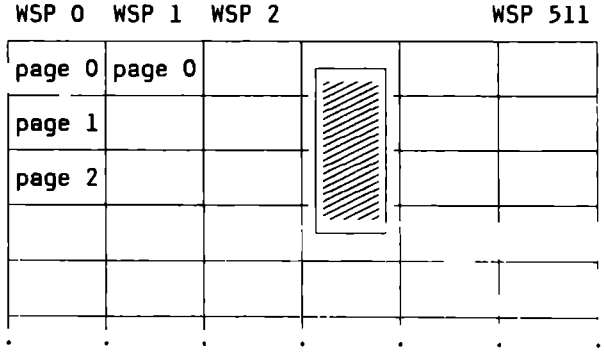
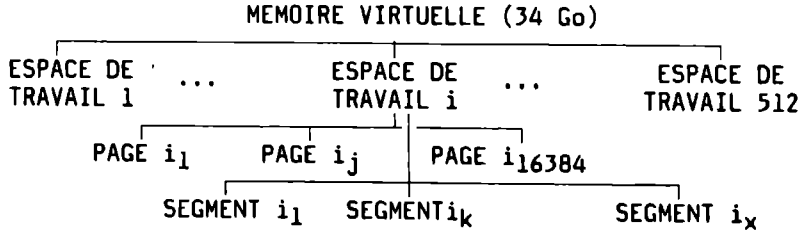


Figure 12r: mémoire virtuelle de GCOS3 découpée en espaces de travail, pages et segments.

segment à cheval sur plusieurs pages

12.9.7 – GCOS8.

A la différence de GCOS3, GCOS8 met en oeuvre l'architecture étendue des DPS8 appelée NSA (New System Architecture). GCOS8 utilise la pagination pour la gestion de la mémoire centrale et la segmentation pour la gestion de la mémoire virtuelle. La mémoire virtuelle globale de 34 Go connaît un double découpage:



- en 512 **espaces de travail** (WSP, Working SPace). Le fait qu'il n'y ait que 512 WSP ne signifie pas que GCOS8 ne gère qu'au plus 512 processus: le même numéro de WSP peut être utilisé par plusieurs processus successivement. Chaque WSP dispose de sa table des pages, tout comme un AS sur MVS, comme le montre la figure 12s. Le WSP est aussi découpé en segments, mais à la différence des autres systèmes, la page n'est pas un sous-ensemble du segment, pour des raisons liées aux capacités. Un segment a une adresse de base définie relativement à un WSP et peut donc débuter n'importe où, à l'intérieur d'une page si nécessaire;

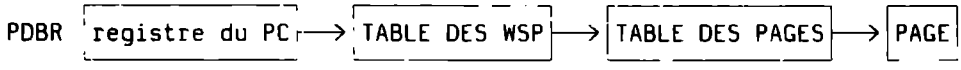
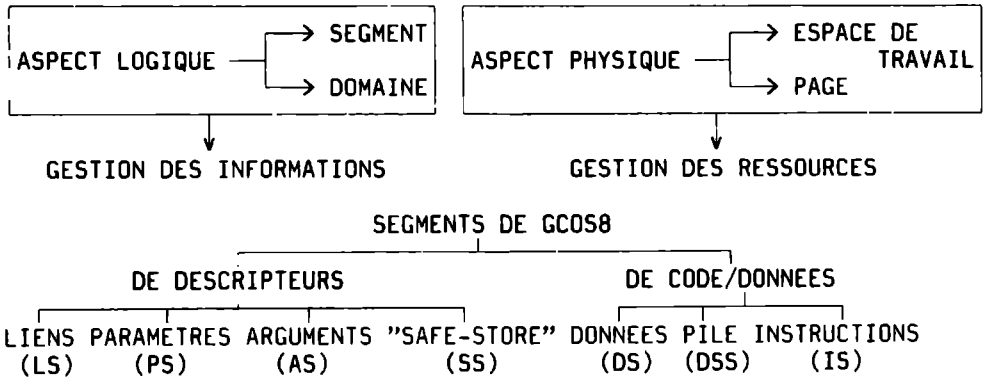


Figure 12s: tables liées aux pages dans GCOS8.

- en **domaines**, chaque domaine est constitué de segments pouvant appartenir à des WSP différents. Le domaine est un ensemble de segments auxquels un processus peut accéder à un moment donné.



chapitre 13

fonctions réseau

Dans le chapitre 10 nous avons associé l'apparition des fonctions réseau au **télétraitement** qui correspondait principalement au fait d'accéder à l'ordinateur depuis un site distant, en utilisant des liaisons du réseau téléphonique (liaisons commutées) ou des liaisons spécialisées (liaison commutée Caducée, liaison permanente, etc.). Les périphériques d'accès, lourds (lecteur de carte, imprimante) pour le traitement par lots, ou légers (clavier/imprimante) pour le temps partagé, étaient déportés à distance (remote) et l'ensemble informatique présentait une structure en **étoile** du point de vue topologique.

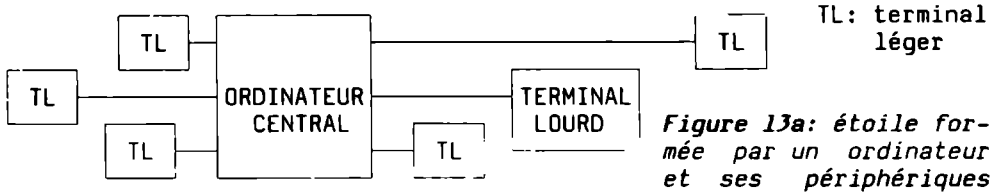


Figure 13a: étoile formée par un ordinateur et ses périphériques distants.

13.1 – Réseaux en étoile.

Au niveau du matériel sont alors apparus l'interface série (correspondant à l'avis V.24 du CCITT) et les modems (modulateurs/démodulateurs) chargés d'adapter le monde numérisé de l'ordinateur à l'univers analogique des lignes de communication de l'époque.

Sur le plan du logiciel sont apparues les **procédures**, les **protocoles** définissant le format des messages échangés et les règles de dialogue entre les périphériques et l'ordinateur central:

- dialogues **asynchrones** dans lesquels les caractères sont transmis indépendamment les uns des autres, utilisés surtout par les terminaux légers dits "asynchrones" et imposant un niveau de présentation (séquences de contrôle ou de commande, séquences "escape") peu développé, du moins au début avec les TTY. On les trouvait surtout dans l'environnement des mini-ordinateurs et du temps partagé;

- dialogues **synchrones** dans lesquels les caractères sont regroupés en **messages** au niveau logique, en **blocs** (ou trame, frame) au niveau physique, pour être transmis. Le nom de la procédure BSC (Binary Synchronous Communications) d'IBM est typique de ce type de protocole. On trouvera VIP chez Bull, les TMM chez CII, etc. Ces procédures correspondent généralement à des familles de procédures, les différents membres de la famille étant associés à des types d'appareils. Il y aura ainsi une BSC pour:
  * les terminaux légers, les écrans claviers de la série 3270;
  * les terminaux lourds comme les 2780, 3780.

Chacune de ces BSC est incompatible avec l'autre. Chez Bull, VIP sera utilisée pour les terminaux légers (interactifs), GRTS pour les terminaux lourds.

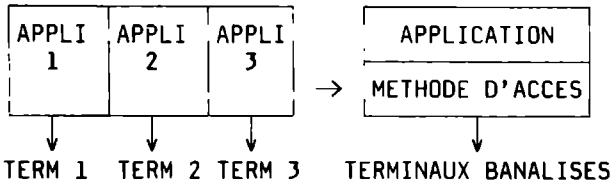


Figure 13b: apparition des SGT.

TERM i: terminaux ne pouvant accéder qu'à l'application APPLI i.

En plus des procédures dites "de ligne" sont apparues, au niveau du logiciel, des méthodes d'accès comme BTAM chez IBM, logiciels que nous appellerons génériquement SGT comme Système de Gestion des Transmissions. Au début les applications géraient elles-mêmes les terminaux distants, puis la notion de méthode d'accès ligne s'est peu à peu dégagée.

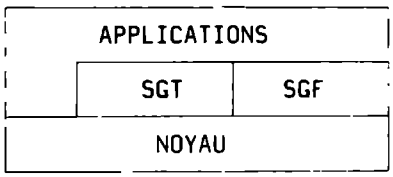
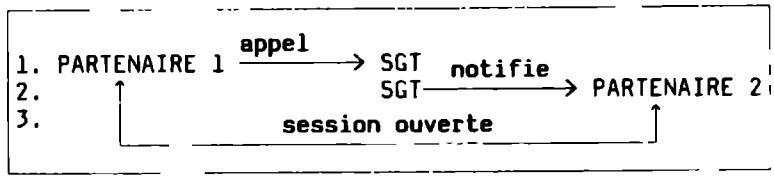


Figure 13c: assimilation des transmissions à la gestion des fichiers.

Le terme de méthode d'accès suggère une assimilation entre fichiers et lignes de communication et effectivement, vue d'une application, une ligne faisait l'objet d'ouvertures, écritures, lectures et fermetures. Avec les mêmes problèmes de partage que pour les fichiers: ainsi sous BTAM deux applications ne peuvent se partager une même ligne et tous les terminaux d'une grappe 3270 (tous reliés à la même ligne) seront obligés de travailler avec la même application, le même travail, la même étape au sens du système d'exploitation. Comme le SGF le SGT gère:

- un répertoire des entités connues du réseau (rôle du SSCP dans VTAM);
- l'accès à ces entités, tout comme le SGF met en relation un processus avec un fichier, le SGT met en relation deux partenaires. L'ouverture d'un dialogue (d'une session) entre deux partenaires

connus du SGT supposera l'intervention du SGT pour la mise en communication (rôle du SSCP dans VTAM):



Sur le plan du matériel, les processeurs frontaux (FEP pour Front End Processeur) sont apparus sur les gros ordinateurs du moment: Datanet de General Electric sur les GE600, puis 3705 d'IBM. Ils ne sont généralement pas autonomes et sont chargés depuis le sous-système central. Ils contiennent les gérants d'appareil, sont responsables de la gestion physique des liaisons, alors que le central se limite à la gestion logique des dialogues. Dans le cas des dialogues asynchrones il n'y a plus interruption du processeur pour chaque caractère, ce qui permet au sous-système central de mieux se consacrer à sa fonction de traitement. IBM conserve cependant le terme de "contrôleur de communications" pour ses frontaux 37X5.

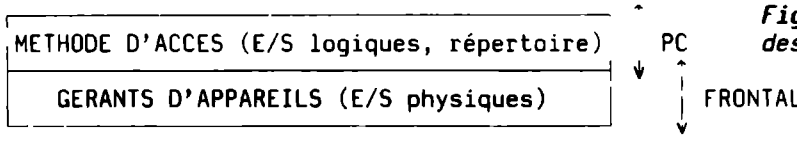


Figure 13d: rôle des frontaux.

13.2 – Evolution hiérarchique.

La structure en étoile s'est peu à peu transformée en une arborescence, avec l'ordinateur central comme racine, de nouveaux équipements s'intercalant entre l'ordinateur central et les terminaux: des concentrateurs/déconcentrateurs gérant les liaisons lentes avec les postes de travail et une liaison rapide avec l'ordinateur central afin de diminuer les coûts de communication dans un premier temps. Comme les processeurs frontaux, ces concentrateurs, souvent des mini-ordinateurs, se sont vus dotés de systèmes d'exploitation temps réel. Ils pouvaient aussi être **téléchargés** (down line loading) à travers les lignes de communication depuis l'ordinateur central. Cette évolution correspond aux réseaux **monodomaines** de SNA, la concentration pouvant être assurée par des frontaux à distance (on dira aussi "déportés").

De nombreux mini-ordinateurs se sont engouffrés dans cette brèche, simulant ("émulant") un terminal lourd ou bien une grappe (cluster) de terminaux interactifs vis-à-vis de l'ordinateur central, offrant une **"intelligence" locale** aux terminaux qui leur étaient directement connectés. Cette évolution correspond, dans un deuxième temps, à une distribution de l'intelligence, ébauche des systèmes distribués.

13.3 – Cas des mini-ordinateurs.

Les minis n'ont pas connu la même évolution hiérarchique que les gros ordinateurs de gestion. La caractéristique de leur évolution en matière de réseau est l'extension du domaine accessible aux utilisateurs d'un ordinateur:

- **connexion logique** ou connexion **à distance**: l'utilisateur physiquement connecté à la machine A déroule en fait une session sur la machine B (figure 13f). Le système A voit un terminal **réel**, le système B un terminal **virtuel**. Lorsque l'architecture réseau incorpore plusieurs machines différentes, il n'est pas évident que toutes soient au même niveau d'avancement dans les fonctions réseau. Ainsi dans le cas de DECnet, jusqu'à DECnet Phase III, les machines A et B devaient avoir le même SE, restriction levée dans la Phase IV. Cette fonction porte parfois les noms de "path thru" (chemin à travers) ou "pass thru". On trouve ainsi VM/PTF (Pass Thru Facility) chez IBM. Cette fonction peut parfois disposer d'émulateurs (au sens de simulation en fait) de terminaux: le terminal relié à A sera par exemple un terminal asynchrone (TTY) connecté à un Mini6 de BULL, alors que la machine B est un ordinateur IBM attendant un écran synchrone du type 3270;

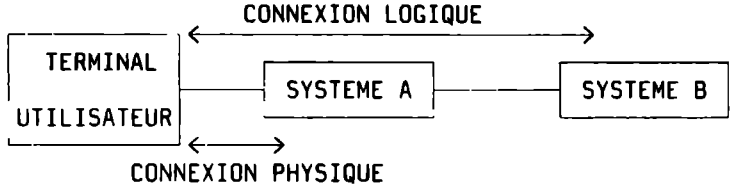


Figure 13e: connexion à distance.

- **accès aux fichiers distants**: l'utilisateur connecté à un système A a accès aux fichiers de toute machine B connecté au système A, pourvu que l'accès "réseau" soit permis par B sur ces fichiers. Les commandes ne sont pas modifiées, seuls les noms des fichiers sont étendus par une composante réseau comme nous l'avons signalé dans le chapitre 7.

Cette évolution est typique des réseaux de DEC (DECnet), de PRIME (Primenet) par exemple.

13.4 – Dialogues entre ordinateurs.

Dans les réseaux hiérarchiques de la section 13.2 il y a toujours un maître, l'ordinateur central, avec son SE qui pilote son étoile ou son arbre. Il dialogue avec des terminaux et gère toutes les ressources. S'il tombe en panne, plus rien ne fonctionne. Ces réseaux sont faits pour permettre le dialogue entre un élément périphérique et l'élément central, non pas pour offrir à tout couple périphérique la possibilité de dialoguer.

L'évolution suivante a conduit à des réseaux dans lesquels on trouve plusieurs ordinateurs centraux, chacun maître de son arborescence ou de son étoile (son "domaine" dans SNA), capables de dialoguer

entre eux. Les modes de fonctionnement peuvent être limités à certains modes comme la soumission de travaux par lots depuis tout point du réseau dans le cas des gros ordinateurs (JES3/NJP, JES2/NJE, NJI avec VM dans le cas d'IBM), l'accès interactif à tout ordinateur du réseau dans le cas des mini-ordinateurs. L'interconnexion de plusieurs ordinateurs, c'est-à-dire la coopération entre plusieurs systèmes d'exploitation se résoud actuellement:

- soit dans le cadre **homogène** de l'architecture réseau d'un constructeur,
- soit dans un contexte **hétérogène** où il s'agit de faire communiquer des machines de constructeurs différents ou des gammes incompatibles entre elles.

13.5 – Achitectures réseau.

Devant le nombre croissant de produits informatiques développés par leurs différents départements, les constructeurs ont rencontré des difficultés pour faire communiquer leurs équipements, d'où une insatisfaction des utilisateurs et l'apparition des architectures réseau des constructeurs. Pour un constructeur, une architecture réseau est un ensemble de formats de messages, de règles de dialogue (des protocoles) que doivent respecter leurs produits afin d'assurer une communication cohérente.

L'architecture réseau constructeur dominante à l'heure actuelle est SNA (System Network Architecture) d'IBM. Au niveau des systèmes d'exploitation elle a fait apparaître un nouveau SGT, VTAM. L'ensemble DOS+VTAM ou MVS+VTAM constitue un tout qui gère un ensemble de ressources formant ce qu'IBM appelle un domaine (domain).

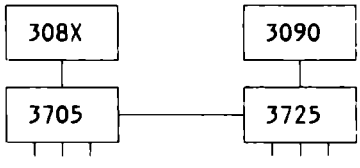


Figure 13f: exemple de réseau SNA multidomaine, avec un domaine géré par le VTAM du 308X, l'autre étant géré par le VTAM du 3090.

C'est VTAM qui attribue les ressources réseau, qui gère les chemins de communication, qui gère les dialogues. Il réalise cette gestion à un niveau logique, entre "unités adressables" (NAU comme Network Addressable Unit), aidé au niveau physique par NCP (Network Control Program) qui tourne dans les frontaux (3705, 3725). Ces entités adressables jouent au niveau réseau le rôle des processus au niveau du SE, des fichiers au niveau du SGF. Sous SNA on aura ainsi les SSCP, le coeur de VTAM, les LU et les PU. Les LU (Logical Unit) représentent aussi bien des applications que des postes de travail: un écran 3270 sera une LU (de type 2 généralement), IMS ou CICS sera lui aussi une LU, et des LU compatibles (comprenant le même langage, les mêmes protocoles) pourront ouvrir des sessions entre elles, session entre application et terminal, session entre deux applications. Notons que contrairement aux architectures réseau que l'on trouve sur les mini-ordinateurs, les fichiers ne sont pas des objets connus au niveau de SNA. Les PU

(Physical Unit) représentent les ressources matérielles du réseau et sont sous le contrôle du SSCP avec lequel elles dialoguent lorsqu'elles sont actives. Toute NAU a un nom logique (nom VTAM, "node VTAM") qui est traduit par le SSCP en une adresse physique, initialement sur 16 bits, puis sur 23 bits (ENA, Extended Network Addressing):

NUMERO DE SECTEUR	NUMERO D'ELEMENT

une adresse SNA est composée d'un numéro de secteur (subarea), d'un numéro d'élément dans ce secteur.

L'évolution suivante a conduit à constituer des ensembles de réseaux SNA interconnectés entre eux par SNI (SNA Network Interconnect), chacun étant géré séparément, tout en permettant des dialogues entre des équipements reliés à différents réseaux. La figure 13g résume la structure d'un réseau SNA actuellement réalisable, hiérarchie sans maître unique.

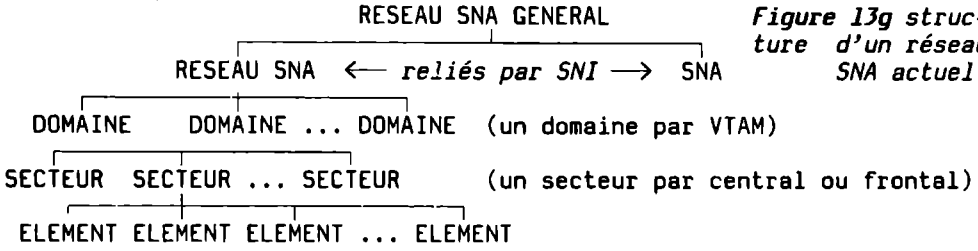


Figure 13g structure d'un réseau SNA actuel.

La liste ci-dessous dresse un inventaire très partiel des architectures réseau des constructeurs. Parmi celles-ci deux s'en dégagent plus nettement: DECnet du second constructeur (place remise en cause par le regroupement de Burroughs et Sperry), DSA de Bull.

- SNA d'IBM suppose l'utilisation des méthodes d'accès VTAM ou TCAM, TCAM étant plus ancienne et moins fréquemment utilisée; elles fonctionnent avec le logiciel NCP dans le frontal. Les anciennes méthodes d'accès réseau BTAM, QTAM, RTAM ne sont pas intégrées dans SNA. Elles peuvent cependant coexister avec TCAM ou VTAM, le logiciel EP (Emulation Program) étant chargé dans le frontal qu'elles utilisent. Un frontal 37X5 peut être partagé par les anciennes et nouvelles méthodes d'accès s'il dispose du logiciel PEP (Partitioned EP);
- DSE/DSA (Distributed System Environment/Distributed System Architecture) chez Bull. Les anciens logiciels réseau GRTS puis NPS des DPS8 ont été remplacés par DNS dans les frontaux Datanet. Les méthodes d'accès pour les applications telles que MDNET n'ont cependant pas été modifiées dans GCOS3, le frontal faisant l'adaptation (rôle de passerelle, gateway) entre ces anciennes méthodes d'accès et DSA. Il en va différemment dans les DPS7, machines de conception française, qui sont plus étroitement intégrées à DSA;
- BNA de Burroughs qui est plus proche d'une architecture réseau de type mini-ordinateur comme DECnet ou Primenet;
- Xodiac de Data General;
- Guardian/Expand de Tandem, basé sur une technique de datagramme;
- Stratanet de Stratus;

- DS de Hewlett-Packard;
- DCA de Sperry;
- Primenet de PRIME;
- DNA de DEC, basé sur une technique de datagramme, avec les produits DECnet, INTERnet, PACKnet;
- Nordnet de Norsk Data;
- Maxnet de Modcomp, etc.

Ces architectures sont généralement présentées sous forme de couches, chaque couche réalisant des **fonctions** bien définies et dialoguant avec les deux couches adjacentes par des **primitives** qui sont indépendantes de la structure interne de ces couches et constituent l'**interface** entre les deux couches correspondantes. D'où une grande indépendance entre couches. Chaque couche fournit un **service** à la couche supérieur, et le service global s'enrichit au fur et à mesure des couches traversées, jusqu'à fournir un service complet de dialogues entre applications et postes de travail, service indépendant des caractéristiques physiques des réseaux sous-jacents. Des couches de même niveau parleront un langage appelé **protocole** de niveau N:

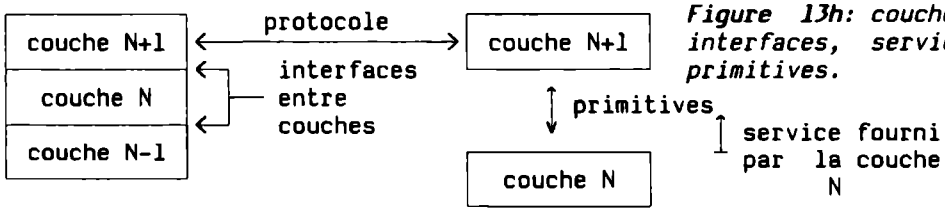


Figure 13h: couches, interfaces, service, primitives.

Dans le cas de MS-DOS les fonctions réseau sont visibles sous forme:

- d'émulateurs qui font du PC un terminal, lourd ou léger, et lui permettent de dialoguer avec d'autres machines, des moyens ou gros ordinateurs en général (PC/mainframe connection);

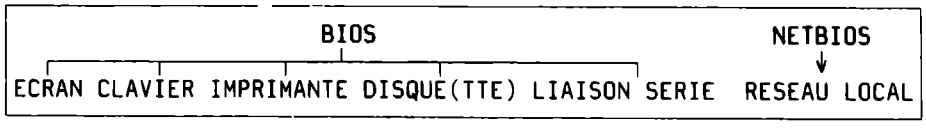


Figure 13i: rôles respectifs du BIOS et du NETBIOS.

- de **NETBIOS**, défini par IBM et Intel, accompagné de NETWORK PROGRAM lorsque le PC est inclus dans un réseau local. Le PC joue un rôle de poste de travail et/ou de serveur si ses ressources locales sont accessibles depuis d'autres postes de travail du réseau local. NETBIOS et NETWORK PROGRAM ne sont bien entendu qu'un cas particulier de produits insérant les PC dans un réseau local.

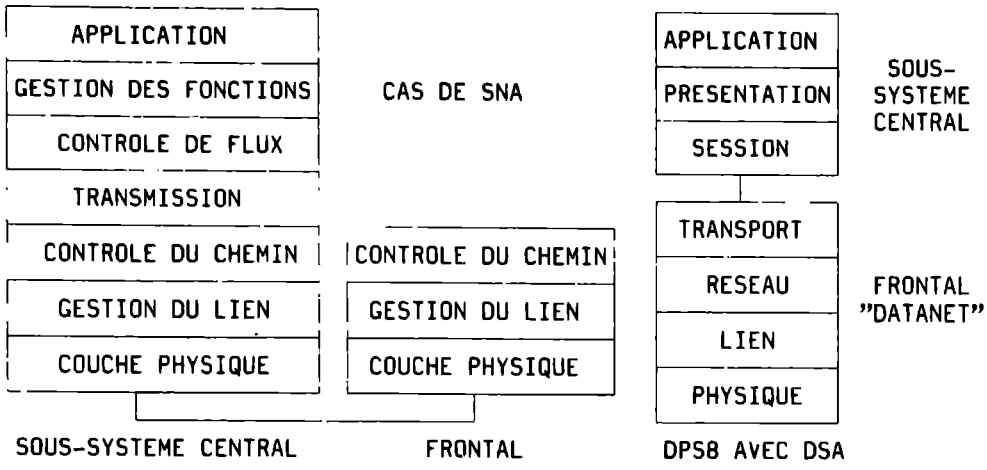


Figure 13j: structures en couches de SNA et DSA.

13.6 – Services offerts.

A l'heure actuelle les architectures réseau offrent les services suivants aux utilisateurs:

- l'accès aux applications quelque soit l'ordinateur sur lequel elles tournent. C'est ce que permettent les "connexions logiques" signalées dans le cas des mini-ordinateurs, VM/PT (Pass Thru), MSNF de SNA, etc.;
- la soumission de travaux depuis tout point du réseau dans le mode traitement par lots: RJP sous JES3, RJE sous JES2 associés aux logiciels du type NJP respectant l'architecture NJE qui permettent aux ordinateurs de s'échanger tout ou partie de certains travaux;
- l'accès aux fichiers distants que nous avons signalé dans le cas des minis, les bases de données distribuées;
- l'accès à d'autres ressources distantes comme des imprimantes, des tables traçantes, des disques, en particulier dans les réseaux locaux;
- le dialogue entre programmes afin de réaliser des applications distribuées: nous avons signalé précédemment le cas du transactionnel distribué, les messageries constituent un autre exemple d'application distribuée suivant des normes propres à un constructeur ou bien des normes internationales comme les avis X.400 du CCITT;
- des applications/transactions peuvent dialoguer en déroulant des sessions entre elles. On parlera de:
  * transactionnel à deux niveaux (Bull). Reprenons un exemple évoqué ci-dessus avec un Mini6 (figure 13e) pour machine A, le terminal dialogue avec le moniteur transactionnel DTF du Mini6; DTF dialogue lui-même avec le transactionnel de la machine B: TDS d'un DPS8 de Bull par exemple;
  * transactionnel coopératif. Chez Bull ceci signifie qu'il y a mise à jour de fichiers sur plusieurs systèmes;

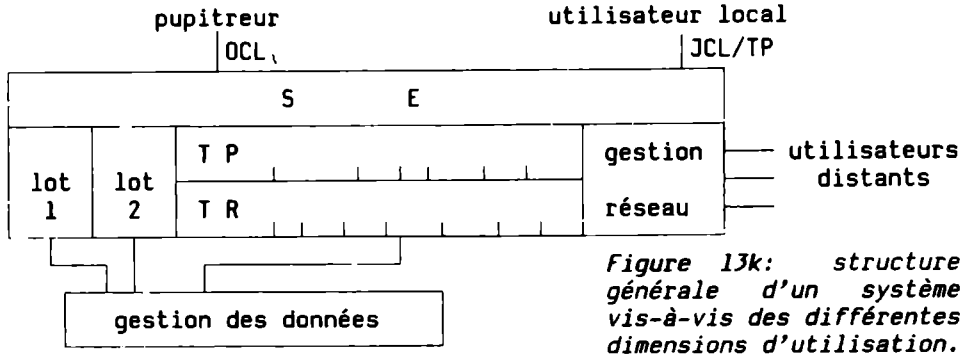


Figure 13k: structure générale d'un système vis-à-vis des différentes dimensions d'utilisation.

- le transfert de fichiers entre ordinateurs. Cette dimension est apparue avec les réseaux reliant entre eux plusieurs ordinateurs. Pendant un certain temps la dimension "traitement par lots" a rempli cette fonction de transferts de fichier: émulations 2780, 3780, HASP dans le monde IBM. Là encore on trouvera des produits aux normes d'un constructeur comme les UFTx de Bull, BDT sur MVS, ou des produits basés sur la normalisation internatioale comme FTAM. Certains produits de transferts de fichiers ressemblent à une messagerie de fichiers.

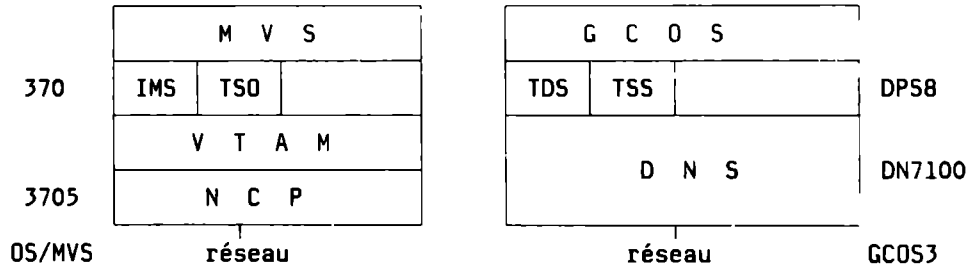


Figure 131: les principaux produits réseau sur les centraux de type 370 ou DPS7.

Pour les responsables des systèmes informatiques les architectures réseau offrent:
- le contrôle des accès;
- la centralisation des alarmes en un point de contrôle;
- le **pupitrage distant** qui permet aux opérateurs d'une machine A de pupitrer tout système qui lui est relié;
- le **téléchargement**, le télévidage (remote dump) des systèmes distants qui fonctionnent ainsi sans opérateurs (unattended mode);
- des outils administratifs permettant d'établir des statistiques.

Dans SNA, NCCF augmenté de produits comme OCCF, NPM, NPDA (bientôt regroupés dans NETVIEW) jouent ce rôle.

En résumé les services offerts tournent autour de l'accès distant et de la distribution des données, des ressources et des programmes.

Comme on peut le voir sur la figure 131, les logiciels implémentant les architectures réseau sont situés:
- presque exclusivement dans le frontal (cas du DN7100, un Mini6 en fait, chez Bull),
- dans le frontal (NCP dans le 3705 chez IBM) et dans le central (VTAM).

couche APPLICATION
couche PRESENTATION
couche SESSION
couche TRANSPORT
couche RESEAU
couche LIAISON
couche PHYSIQUE

Figure 13m: modèle de référence de l'ISO.

Les architectures réseau des constructeurs informatiques sont plus ou moins proches de ce modèle, la tendance étant de s'en rapprocher.

Pilotée par les gérants d'appareil qui se trouvent dans le GESP.

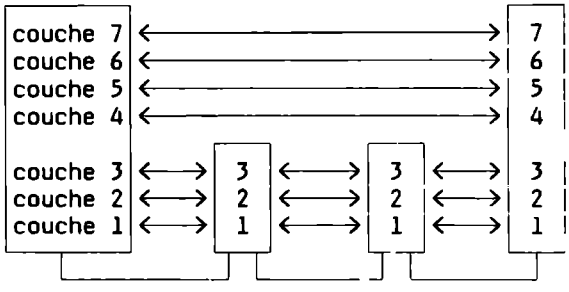
13.7 – Modèle ISO.

Nous avons dit plus haut qu'il y avait deux grands types d'architecture de réseau:
- celles basées sur des produits **homogènes**, ou d'une certaine homogénéité; c'est le cas des architectures des constructeurs classiques;
- celles qui transcendent les modèles et systèmes d'exploitation et veulent rendre communicants des produits **hétérogènes**, les rendre **ouverts** au dialogue généralisé. C'est le but des travaux appelés OSI (Open System Interconnection, Interconnexion des Systèmes Ouverts) menés par l'ISO (International Standard Organization) qui est un organisme international définissant des normes.

L'ISO ne conçoit pas de produits comme les constructeurs, mais publie des définitions d'**interface** (définition du service offert et accès à ce service) et de **protocoles** dans le cadre d'un modèle en couches dit modèle de **référence** des "systèmes ouverts". Ce modèle définit sept couches dans lesquelles sont classées toutes les fonctions mises en oeuvre pour que deux interlocuteurs puissent dialoguer de façon convenable:
- un utilisateur dialoguant avec une application à l'aide d'un écran-clavier par exemple,
- deux applications dialoguant entre elles.

Un système parlant les protocoles ISO sera dit ouvert et pourra dialoguer avec tous les systèmes ouverts. L'ISO distingue deux grands types de système:

- les **systèmes d'extrémité** (End System) constitués par les ordinateurs ou les postes de travail. Ils disposent des 6 ou 7 couches et ont pour but de traiter l'information;
- les **systèmes intermédiaires**, aussi appelés relais, passerelles (gateway, bridge) dont la fonction n'est pas de traiter l'information, mais de la transporter en l'adaptant si nécessaire. Ils ne disposent que des couches basses du modèle de référence. Les commutateurs de paquets du réseau Transpac sont des systèmes intermédiaires typiques disposant des 3 couches basses, et vérifiant la norme X.25 du CCITT. Les frontaux 37X5 d'IBM sont aussi des systèmes intermédiaires.



Exemple de deux systèmes d'extrémité reliés par deux systèmes intermédiaire ne disposant que des 3 premières couches. Les flèches représentent les dialogues entre les 4 systèmes ainsi interconnectés.

Les travaux de l'ISO au niveau des systèmes d'extrémité, du CCITT au niveau des systèmes intermédiaires et des services publics (messagerie par exemple), ont une incidence sur le développement des produits réseau des constructeurs. Tous ont maintenant des logiciels leur permettant de se raccorder sur des réseaux à commutation de paquets suivant la norme X.25. Peu à peu ils implémentent les protocoles des couches supérieures du modèle de l'ISO: OTSS d'IBM qui monte jusqu'à la couche 5, la prise ISO/DSA (PID) de DSA, les messageries X.400, le transfert de fichier FTAM, etc. Intel offre actuellement un ensemble de produits destinés au réseau local industriel MAP et structurés conformément à cette normalisation:

MAP-NET	couches 5 (ISO 8327, noyau), 6 (vide), 7 (FTAM)
iNA 960	couches 2b, 3 (ISO 8473, Internet), 4 (ISO 8073 classe 4)
SXM554	carte réalisant les fonctions des couches 1 à 2a

On assiste en fait à une double convergence sur les normes SNA et ISO: en plus de sa propre architecture, chaque constructeur implémente tout ou partie des protocoles ISO et SNA: émulation de grappes de terminaux (PU2/LU2), de terminaux lourds 3770, de frontaux même (projet JANUS de Bull devenu OSF).

chapitre 14

partage et protection

14.1 – Le problème posé.

Que doit protéger et partager le système? Il doit **protéger**:
- l'accès au système lui-même, seuls les utilisateurs autorisés ont le droit de faire appel à ses services;
- les objets qu'il gère, les utilisateurs qu'il sert, contre les accès non autorisés;
- l'ensemble du système contre les incidents dus à la fiabilité toujours limitée des ressources physiques utilisées, ou provoqués par des utilisateurs mal intentionnés.

Il doit **partager** les ressources entre les différents utilisateurs simultanément actifs.

Nous ne traiterons pas dans ce chapitre des problèmes de sécurité posés par les incidents ou externes au système d'exploitation. Seul un sous-ensemble du problème général présenté dans [TUR76], [SHA77] par exemple sera donc traité ici.

14.2 – Modèle général.

Pour introduire les différents aspects de la sécurité nous utiliserons le modèle présenté figure 14a, modèle établi à la fin des années 60 et référencé par [VAX84A], [AME83]. Un système informatique y est décrit en termes de **sujets** actifs, d'**objets**, de base des autorisations, d'un journal d'audit et d'un mécanisme mettant en oeuvre le système de sécurité qui:
- intervient dès qu'un sujet accède à un objet. Il doit être "complet", couvrir tous les accès;
- utilise une base et produit une trace protégées contre les lectures et modifications non autorisées;
- est réalisé de façon suffisamment compacte, simple et bien structurée pour établir efficacement les bases de la sécurité.

Ce mécanisme de base est généralement réalisé par du matériel et du logiciel appelé "noyau sécurisé".

Les droits (la base des autorisations) sont représentables sous forme de matrice, avec les sujets en ligne et les objets en colonne, et dans chaque case les droits du sujet sur l'objet correspondant.

	objet y
sujet x	droits

Pour des raisons d'ordre pratique liées surtout à la taille de la matrice, la base des autorisations n'est jamais implémentée sous cette forme de matrice comme nous le verrons par la suite.

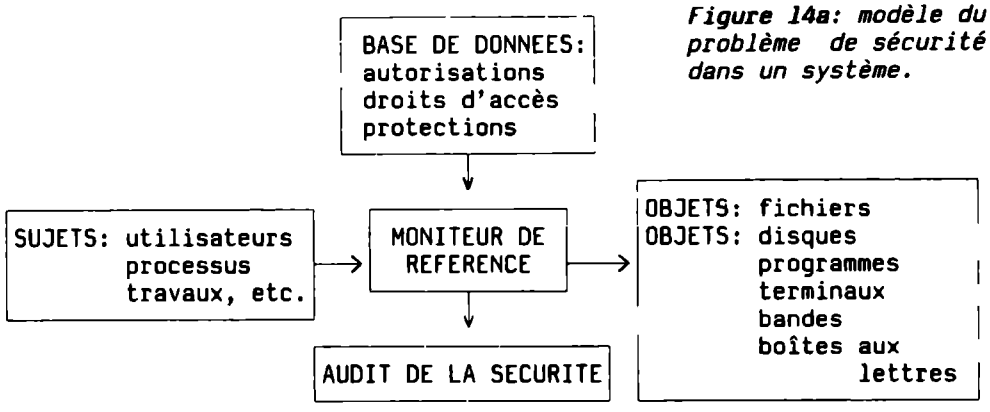


Figure 14a: modèle du problème de sécurité dans un système.

14.3 – Protection de l'accès au système.

Le contrôle de l'accès au système est basé sur des moyens logiques comme l'association entre un nom d'utilisateur et un ou plusieurs **mots de passe**. Il peut aussi comporter des moyens physiques comme des clés, des badges magnétiques. Dans les systèmes interactifs l'adresse du terminal peut aussi être utilisée pour restreindre l'accès, un mot de passe supplémentaire étant alors demandé sur certains terminaux. La connexion peut aussi mettre automatiquement en relation l'utilisateur avec une application bien définie et non pas lui donner toute la puissance des commandes du SE; la session peut encore être établie par l'application. Afin d'augmenter la sécurité offerte par les mots de passe, les systèmes utilisent les mécanismes suivants:

- ils forcent l'utilisateur à se doter de mots de passe suffisamment longs pour qu'ils soient difficiles à deviner et longs à fabriquer automatiquement. Le "qualifieur" /PWDMINIMUM de AUTHORIZE de VMS joue ce rôle;
- le nombre d'essais successifs permis est limité en cas d'erreur de frappe du mot de passe afin d'en empêcher la fabrication automatique; de même le temps entre deux essais consécutifs. L'opérateur ou l'administrateur de la sécurité est généralement averti de ce genre d'incident, c'est la fonction du journal d'audit dans le modèle de la figure 14a;
- lors d'une connexion réussie l'utilisateur est averti que l'on a éventuellement essayé de se faire passer pour lui, il peut alors vérifier la date de sa dernière connexion;
- l'utilisateur est forcé de modifier ses mots de passe périodiquement. C'est ce que permet de définir le qualifieur /PWDLIFETIME de AUTHORIZE de VMS;

- dans le système lui-même, les mots de passe choisis par les utilisateurs sont cryptés afin que le personnel gérant l'ordinateur ne puisse pas profiter de ses privilèges pour les lire: c'est le cas de VMS où l'algorithme de codage des mots de passe n'a pas de réciproque; dans TSO par contre les mots de passe figurent en clair dans le fichier SYS1.UADS ([CON86]). RACF résoud ce problème dans l'environnement IBM;
- la génération de mots de passe par le système pour éviter que l'utilisateur ne choisisse des mots simplistes: nom, prénom, initiales, etc. La commande SET PASSWORD/GENERATE=n de VMS offre ainsi un mot de passe d'au moins n caractères;
- rappel par le système une fois la session établie: le système coupe la session puis rappelle l'utilisateur.

Certains systèmes (VMS, Multics) permettent de restreindre les jours et heures où un utilisateur donné a accès au système (shift restriction).

Les connexions à distance (remote login, voir chapitre 13) ou les accès à des ressources distantes (fichiers par exemple) posent des problèmes supplémentaires puisqu'il y a transfert de mots de passe entre ordinateurs. VMS offre une solution appelée "proxy login" (cf. [VAX84A]).

14.4 – Modes de fonctionnement.

Une fois l'utilisateur reconnu et accepté, il s'agit de limiter ses activités, ou plus exactement celles de ses **agents** que sont les processus qu'il déclenche, à celles auxquelles il a droit et qui sont associées à son nom de connexion. Le mécanisme de base sur lequel sont construites toutes les protections est assuré par le matériel, c'est le **mode de fonctionnement** introduit dans le tome I. Il a deux effets principaux:

- le premier conduit à décomposer le jeu d'instructions en plusieurs sous-ensembles, au minimum deux: en mode "privilégié" (maître, système, etc.) toutes les instructions sont exécutables par le matériel. Dans le mode non privilégié (esclave, problème, etc.) seule une partie des instructions machine est exécutable et toute tentative d'exécuter une instruction interdite conduit à une interruption;
- le second effet limite la portion de mémoire accessible puisque l'accès à une zone de mémoire peut être conditionné par le mode de fonctionnement. On trouve généralement ceci dans les machines avec anneaux, c'est-à-dire avec plus de deux modes de fonctionnement.

La partie sécurisée (un sous-ensemble du noyau, des processus privilégiés) du SE tourne ainsi en mode privilégié et a tous les droits, c'est elle qui contrôle tous les accès des sujets vers les objets et joue ainsi le rôle du moniteur de référence de la figure 14a. Tous les processeurs n'ont pas nécessairement cette notion de mode et dans ce cas il n'y a aucune protection possible: cas des 6502, des Z80, de l'ensemble des processeurs 8 bits, de la majorité des 16 bits comme le 8086.

Les instructions privilégiées sont celles qui permettent de réa-

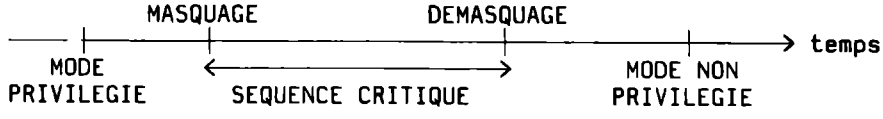
liser des entrées/sorties, de manipuler des registres utilisés pour gérer la machine (cas du registre de base dans un DPS8, des registres de contrôle CRi dans un 370), de modifier les protections des zones mémoire (cas primordial dans les machines qui n'ont pas d'instructions d'entrée/sortie, voir tome III), d'accéder aux tables des pages, tables des segments, d'inhiber ou activer les interruptions. L'iAPX386 qui dispose de quatre modes de fonctionnement (4 anneaux) permet de définir quels anneaux ont droit aux instructions d'E/S (champ IOPL du registre EFLAGS). A tout moment le mode de fonctionnement est défini dans le mot d'état de la machine (PSW sur 370, les deux bits de droite du sélecteur CS sur iAPX286/386).

MACHINE/SYSTEME	NOMBRE d'ANNEAUX	
DPS7	4	
VAX	4	k/e/s/u
PRIME	4	
Multics	8	
ICL 2900	16	
MV/X000	8	
NORSK DATA	4	
iAPX286	4	k/s/x/a
iAPX386	4	
WE32000	4	k/e/s/u
NOS/VE	64	

Tableau 14b: exemples de de machines à anneaux.

k: noyau (kernel)
e: exécutif
s: superviseur
x: extension du SE
a: application
u: utilisateur

Le fait que les interruptions soient gérées par le SE constitue aussi un élément de sécurité puisqu'elles font passer la machine en mode privilégié (cf. tome III pour les interruptions). Le fait qu'elles puissent être masquées est aussi un élément essentiel pour la réalisation de **séquences critiques**, c'est-à-dire d'ensembles d'instructions dont on est certain que **rien** ne viendra interrompre l'enchaînement. A la différence des verrous que nous verrons plus loin, et qui ne créent des sections critiques que dans la mesure où chacun respecte la procédure, le masquage des interruptions produit des séquences critiques absolues, qu'aucun mauvais joueur ne pourra interrompre:



14.5 – Cas des multiprocesseurs.

Si le mode de fonctionnement permet de créer des mécanismes de protection et de sécurité dans la mesure où un pilote unique prend toutes les décisions, il n'en va plus de même sur un multiprocesseur puisque plusieurs processeurs peuvent dérouler simultanément les mêmes séquences du pilote sécurisé, chaque processeur étant en mode maître. Prenons l'exemple de [LUN82] concernant le vol de page sous MVS, défini dans le chapitre 11: l'algorithme de vol cherche une page à

vider dans la table des cadres (PFT); pour chacune des pages il examine si elle est fixée en mémoire centrale. Supposons qu'il trouve une page non fixée et qu'il la "vole" alors qu'au même instant la page est fixée par du code se déroulant dans un autre processeur: le système va ainsi se trouver dans un état incohérent. Dans les premiers multiprocesseurs d'IBM (avant MVS), ce problème était réglé en s'assurant qu'un seul processeur était en mode privilégié, le processeur "maître" dans une configuration asymétrique (cf. d'autres exemples dans le tome I), mais le débit global s'en ressentait. Par la suite IBM a introduit avec les 370 des instructions de type "Test and Set" (T&S, cf. tome I) comme CS (Compare and Swap) ou CSD (CS Double word): CS lit un mot mémoire, le compare avec le contenu d'un registre et selon qu'il y a égalité ou différence remplace le mot lu par le contenu d'un autre registre, réalisant ainsi une généralisation de l'instruction plus courante "test and set" (TS sur 360/370) puisque le mot lu n'est pas accessible pour un autre processeur tant que CS n'est pas terminée.

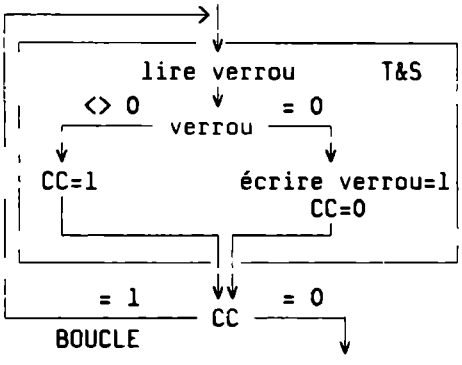


Figure 14c: attente active à base d'intruction du type "test and set".

CC: Code Condition ou Indicateurs

La boucle est en dehors de l'instruction T&S bien entendu.

L'algorithme de "vol" de MVS pourrait utiliser une instruction CS au niveau de chaque entrée analysée de la PFT, mais le coût en est estimé trop élevé et la solution retenue est un mécanisme de verrou: un mot mémoire appelé SALLOC donne accès à la PFT: tout processeur voulant accéder à la PFT regarde d'abord le contenu de SALLOC par une instruction CS. Si ce contenu indique que la PFT est disponible, CS modifie SALLOC pour qu'il indique que la PFT est maintenant occupée; sinon le code condition (cf. tome I) de CS indique que la PFT est occupée et le processeur attendra qu'elle se libère en bouclant sur l'instruction CS ([CRO75] parlera d'**attente active**, MVS de "spin lock", figure 14c). Cette boucle ne se justifie que si elle dure peu, d'où le nombre croissant de tels verrous dans les différentes versions des OS d'IBM. Dans MVS/370 la boucle est en fait de type "suspend" (figure 14m), elle est de type "spin lock" dans MVS/XA avec quatre verrous au lieu d'un ([DUB86]).

Un mot tel que SALLOC est appelé un **verrou** (lock) chez IBM, DEC entre autres. GCOS3 utilise une technique analogue basée sur des verrous appelés "guichets" (gate, grille, guichet dans [CRO85]) et

l'attente active sera nommée "gating"; l'instruction de support y est une "test and set". Notons que ce genre d'instruction s'exécute toujours sur un mot de la mémoire centrale dans un DPS8, jamais dans l'antémémoire. Le préfixe LOCK des instructions du 8086 permet de construire une solution analogue.

En conclusion le support du partage et de la protection dans les multiprocesseurs est l'instruction de type "**test and set**", avec des conventions du SE qui créent des verrous au moyen de cases mémoire accédées par ce type d'instruction. Un acteur ne respectant pas la procédure du verrou pourrait très bien manipuler la structure protégée par le verrou et provoquer des incohérences. D'où le besoin d'autres mécanismes, comme les anneaux, pour protéger l'accès aux verrous eux-mêmes afin que seuls des logiciels "sécurisés" puissent les accéder.

Les verrous sont aussi utilisés sur des monoprocesseurs pour protéger l'accès à des ressources non partageables, mais sans attente active.

14.6 – Protection en mémoire.

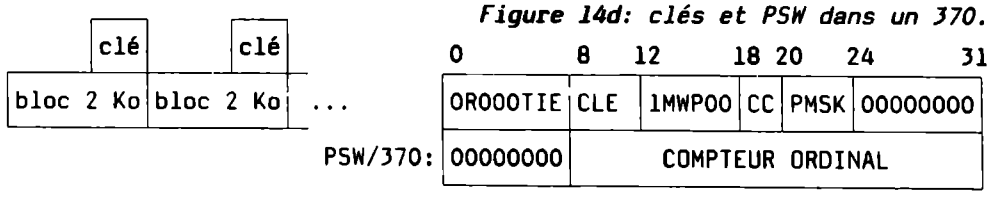
Les objets à protéger se trouvent en mémoire centrale, en mémoire virtuelle, ou sur mémoire périphérique. Nous traiterons d'abord des deux premières mémoires, les objets concernés étant le code, les informations, les descripteurs du code ou des informations. Le code est sujet aux opérations de chargement, exécution (appel) et lecture (copie), les informations et les descripteurs peuvent être créés, modifiés et lus, voire transmis. Il faut encore ajouter l'opération de modification des protections et le tableau est dressé.

Les mécanismes utilisés pour protéger l'information en mémoire reposent sur les dispositifs tels que registres de base privilégiés, clés, bits d'écriture, tables de pages, anneaux, segmentation, capacités, machine virtuelle.

14.6.1 – Registres, clés et bits.

Les **registres de base privilégiés** permettent de confiner le code utilisateur entre des frontières infranchissables: cas des deux registres base et limite des DPS8 qui définissent ainsi une partition d'où l'utilisateur ne peut s'échapper. Sur les 1100 la partition est généralisée à plusieurs "bancs" protégés par autant de couples [RB,RL].

Figure 14d: clés et PSW dans un 370.



Les **clés** sont utilisées dans les systèmes IBM360 et 370 où la mémoire est découpée en blocs de 2 ou 4 Ko, chaque bloc étant associé à une clé de n bits. Lorsqu'un processus se voit attribuer de la

mémoire:
* les blocs de la mémoire centrale alloués prennent la clé de 4 bits c associée au processus,
* le champ clé (key) du mot d'état de la machine (PSW) contient cette valeur c lorsque le processus prend le contrôle du processeur, lorsqu'il est actif.

A chaque référence à la mémoire centrale produite par le processus au cours de son exécution, le matériel compare automatiquement:
* la clé C du bloc de mémoire centrale accédé,
* avec la clé c du mot d'état.

Si les deux clés sont égales (c=C), l'accès est permis, sinon une interruption pour violation d'accès MC est générée. Une valeur particulière de la clé (zéro sur IBM370) permet au superviseur d'accéder, sans contrainte, à toute zone MC. Ce contrôle est systématique pour les opérations d'écriture, il est mis en oeuvre en fonction du bit F (fetch) de la clé pour les opérations de lecture. Sous MVS, malgré la pagination (protection transversale entre utilisateurs selon [DUB86]) et l'augmentation du nombre de travaux simultanés, les clés conservent leur intérêt et leurs valeurs sont utilisées comme suit (protection longitudinale entre AS et SE):

0	: SCP (System Control Program, le noyau)
1	: JES
5	: gestion des données (VSAM)
6	: TCAM ou VTAM
7	: IMS
8	: utilisateurs V=V
9-15	: utilisateurs V=R

Dans certains ordinateurs (MITRA15 par exemple) un bit supplémentaire est stocké avec chaque mot MC. Selon que ce bit vaudra un ou zéro, on pourra ou non écrire dans le mot. Cette protection est systématique, que l'accès soit fait en mode privilégié ou non privilégié. Elle permet de construire des portions de mémoire morte en mémoire centrale.

PTE

NUMERO DU CADRE		00	D	A	00	U/S	R/W	P

Figure 14e: protection par pagination dans l'iAPX386.

	U/S	R/W	ANNEAU 3	ANNEAUX 0-2
	0	0	-	L/E
- : aucun accès	0	1	-	L/E
L : lecture seulement	1	0	L	L/E
L/E: lecture/écriture permises	1	1	L/E	L/E

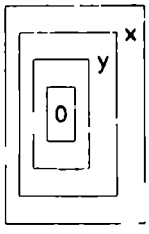
14.6.2 – Pagination.

La pagination introduit une protection entre utilisateurs dans la mesure où elle limite l'accès aux seules pages définies dans les tables de pages du processus actif. Si deux processus ont des tables de pages différentes ils accéderont automatiquement à des zones de mémoire différentes, sauf si l'on a volontairement décidé qu'ils ont

des zones communes. A l'intérieur de l'espace adressable par le processus certaines zones peuvent lui être interdites, par le mécanisme des anneaux par exemple. Ce qui permet de protéger des blocs gérés par le SE pour le compte du processus.

14.6.3 – Pagination et anneaux.

Le système des anneaux (ring) doit être vu comme une généralisation des modes maître et esclave qui correspondent à 2 anneaux.



L'idée générale de l'anneau est de créer un ensemble de modes de fonctionnement strictement ordonnés: les droits ou privilèges d'un anneau **x** seront inférieurs ou égaux à ceux d'un anneau **y** plus prioritaire. Les anneaux sont généralement numérotés et zéro représente l'anneau le plus prioritaire. Dans le cas précédent on aura x > y. Plus le numéro est élevé, moins l'anneau est prioritaire, d'où un aspect parfois trompeur dans les exemples.

Dans l'**iAPX386** qui dispose de la pagination et de la segmentation les anneaux n'interviennent pas directement dans la pagination. Les deux bits U/S (User/Supervisory) et R/W (Read/Write) d'une entrée de la table des pages sont utilisés en fonction de deux modes de fonctionnement: privilégié (anneaux de numéro 0 à 2) et non privilégié (anneau numéro 3), le numéro d'anneau étant défini au niveau du descripteur de segment (DPL). Comme il y a une pagination à deux niveaux, la plus restrictive des deux protections est prise en compte. Un processus utilisateur s'exécutant dans l'anneau 3 ne pourra pas lire une page avec les bits U/S égaux à zéro, il pourra par contre lire une page avec le bit U/S à un. Notons qu'actuellement iRMX386 ne met pas en oeuvre ces protections, tout s'exécutant dans l'anneau zéro.

Les pages du **VAX** sont protégées par un mécanisme de "modes d'accès", un code de protection P étant stocké dans chaque entrée de la table des pages (PTE):

31	30	26	25	21	20	0
V	P	M	inutilisé		numéro de cadre (PFN)	

V: validité
M: modification
P: protection (4 bits)

CODE P	MODE D'ACCES K	E	S	U	CODE P	MODE D'ACCES K	E	S	U
0	-	-	-	-	8	LE	LE	LE	-
1	r	r	r	r	9	LE	LE	L	-
2	LE	-	-	-	10	LE	L	L	-
3	L	-	-	-	11	L	L	L	-
4	LE	LE	LE	LE	12	LE	LE	LE	L
5	LE	LE	-	-	13	LE	LE	L	L
6	LE	L	-	-	14	LE	L	L	L
7	L	L	-	-	15	L	L	L	L

Valeurs du champ P dans le VAX-11/780:

- : aucun accès
r : réservé (résultat inconnu)
L : lecture seulement
LE: lecture et écriture

Un mécanisme strict d'anneau, différent de ce qui est réalisé et sur

le VAX et sur l'iAPX386, conduirait à une table différente. Les 4 bits du champ P y seraient décomposés en deux fois 2 bits, 2 pour la lecture (bits de gauche) et 2 pour l'écriture (bits de droite):

	ANNEAU					ANNEAU			
CODE P	0	1	2	3	CODE P	0	1	2	3
0	LE	-	-	-	8	LE	L	L	-
1	LE	E	-	-	9	LE	LE	L	-
2	LE	E	E	-	10	LE	LE	LE	-
3	LE	E	E	E	11	LE	LE	LE	E
4	LE	L	-	-	12	LE	L	L	L
5	LE	LE	-	-	13	LE	LE	L	L
6	LE	LE	E	-	14	LE	LE	LE	L
7	LE	LE	E	E	15	LE	LE	LE	LE

Valeurs du champ P avec des anneaux stricts.

- : aucun accès
L : lecture seulement
LE: lecture et écriture

Comme dans le cas du VAX, les droits d'un anneau numéro x sont inférieurs à ceux d'un anneau y<x, mais on ne sait pas y définir des zones inaccessibles au noyau (K ou anneau 0), accessibles en lecture seulement (P=3 sur VAX); il y a de plus de nombreux cas où l'on a le droit d'écrire, mais pas de lire, situation rarement désirée. On notera qu'il n'y a pas de contrôle pour l'exécution dans le VAX au niveau de la pagination, pages de code et de données ne faisant pas l'objet d'une différentiation; le contrôle de l'exécution ne se fait qu'au niveau des fichiers. Les bits 21 à 25, inutilisés par le matériel, servent à VMS:

- pour stocker (bits 23 et 24) le mode d'accès permettant de modifier la protection P ou supprimer la page; si le contrôle des opérations de lecture, écriture et exécution sont fréquemment implémentées au niveau du matériel, les opérations de création, destruction et changement des protections le sont plus généralement dans le logiciel;
- pour initialiser la page avec des zéros, la partager, noter les transitions d'état, etc. (bits 21 et 22).

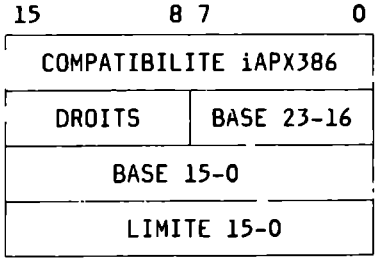
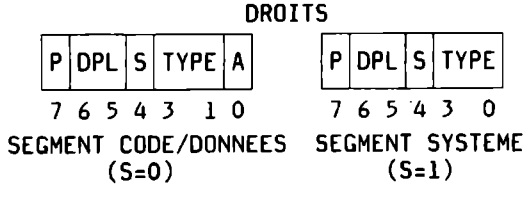


Figure 14f: format d'un descripteur de segment de l'iAPX286.



14.6.4 – Segmentation et anneaux.

Nous avons utilisé l'iAPX386 pour illustrer la pagination et nous pourrions le conserver pour la segmentation puisque ce microprocesseur dispose des deux mécanismes, mais l'iAPX286 lui étant compatible et actuellement plus répandu, nous le prendrons à témoin dans les figures 14f à 14j. La figure 14f présente le descripteur de segment qui occupe

8 octets découpés en 4 zones:
- 24 bits pour l'adresse du segment en mémoire centrale (BASE 0-23);
- la longueur courante du segment sur 16 bits (LIMITE 0-15);
- son état (bit P de présence ou non en mémoire centrale) et ses droits d'accès;
- ainsi que 16 bits de compatibilité avec l'iAPX386.

La figure 14g donne la typologie des segments telle qu'elle se déduit du champ TYPE de l'octet des droits d'accès, avec une grande diversité dans les descripteurs dits "système".

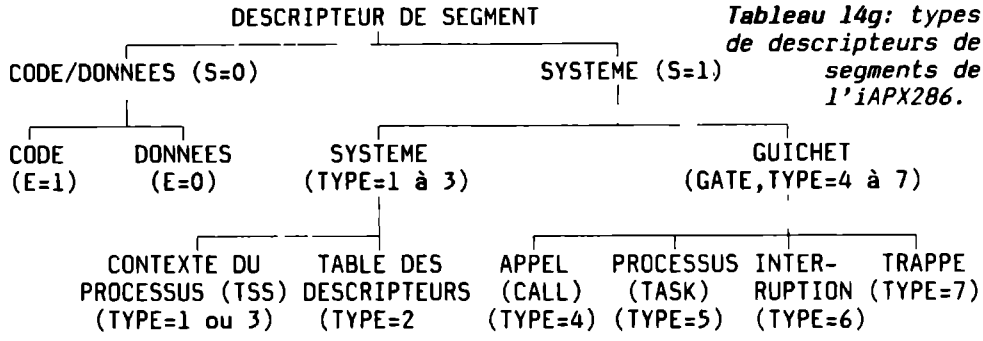
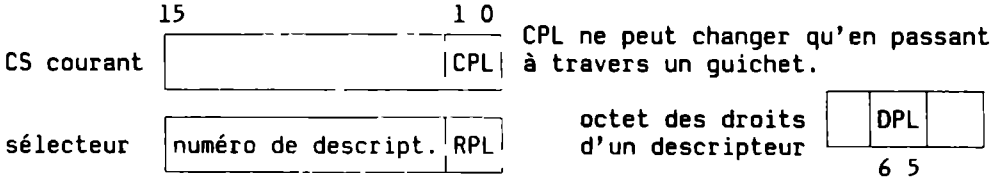


Tableau 14g: types de descripteurs de segments de l'iAPX286.

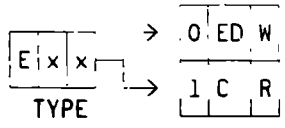
Avant d'aller plus loin arrêtons-nous quelques instants sur les formes que prennent les anneaux dans la segmentation de l'iAPX:
- CPL est l'anneau courant, celui dans lequel s'exécute le processus actif, tel que défini par les deux bits de droite du sélecteur CS;
- RPL est le numéro d'anneau qui figure dans les deux bits de droite des autres sélecteurs comme CS, ES;
- DPL est le numéro d'anneau qui se trouve dans un descripteur de segment, c'est-à-dire dans une entrée d'une table de segments:



Les instructions déroulées par le processus courant référencent des informations et du code, "directement" par les champs adresse logique (AL) des instructions, ou "indirectement" par des descripteurs.

La figure 14h détaille le cas des descripteurs de code ou de données dans lesquels on remarque un certain nombre d'éléments de protection:
- contre l'**écriture** (bit W) pour les segments de données;
- contre la **lecture** pour les segments de code: si R est égal à 1, le code peut être chargé en mémoire, mais il ne peut pas être lu, donc pas copié. En aucun cas on ne peut écrire dans un segment de code, d'où l'absence de bit W;

- contre l'**exécution** de code insuffisamment sûr appelé par JMP, CALL, INT: l'appelant s'exécute dans l'anneau CPL et ne peut en aucun cas appeler du code ayant un DPL supérieur, c'est-à-dire dans un anneau moins privilégié. L'appel de code plus privilégié (DPL<CPL) est possible si le bit C (bit de conformité) de l'appelé est égal à un; le niveau de privilège reste cependant inchangé, CPL conservant la même valeur;
- contre l'accès à des informations par du code insuffisamment privilégié: le maximum de CPL et RPL doit être inférieur ou égal à DPL: on ne peut accéder à des données dans un anneau plus privilégié.



SEGMENT DE DONNEES: ED: direction d'expansion
W : écriture permise/non
SEGMENT DE CODE: C: conformité
R: lecture permise/non

Figure 14h: descripteurs de segment contenant du code ou des données.

Le cas des appels en passant par des descripteurs, donc en changeant de segment ou même de processus est plus complexe que l'accès aux données. L'appel par un descripteur de code ne peut en aucun cas modifier le privilège, c'est-à-dire CPL. Les descripteurs système (guichet) par contre offrent un mécanisme d'appel contrôlé avec possibilité d'augmentation du privilège courant, la nouvelle séquence de code se trouvant dans un autre segment, ou même dans un autre processus. Le tableau 14i tiré de [INT85D] résume les différents types d'appel et fait apparaître 3 grands mécanismes de transfert, rupture de séquence:

	DESCRIPTEUR CODE	DESCRIPTEUR SYSTEME
DPL > CPL	rejet	rejet
DPL = CPL	ok	ok
DPL < CPL (1)	ok si C=1 CPL inchangé	ok CPL=DPL

Figure 14i: appels en passant par des descripteurs dans l'iAPX286.

(1) par CALL, il ne peut y avoir de changement de privilège par JMP.

- l'instruction d'**appel** CALL FAR qui banalise appel des procédures et activation des processus, et permet de changer de privilège (d'anneau). Avec un guichet d'appel il faut que le maximum de CPL et de RPL soit inférieur ou égal au DPL du guichet, le privilège final (nouvelle valeur de CPL) étant inférieur ou égal au DPL du segment exécutable;
- l'instruction INT et les **interruptions** qui sont ainsi assimilées à des instructions d'appel, tendance déjà observée sur des microprocesseurs comme le 6502 ou le 8086. Elles suivent les mêmes règles

que l'appel par l'instruction CALL et peuvent aussi provoquer une commutation de processus;
- l'instruction de saut JMP FAR, saut dans un autre segment ou activation de processus dans le cas d'un descripteur TSS, mais sans changement de privilège; il faut que le maximum de CPL et de RPL soit inférieur ou égal au DPL du guichet (max[CPL,RPL]<=DPL[guichet]), sauf si le segment est "conforme".

Les instructions de retour RET et IRET jouent dans le sens inverse et ne peuvent pas augmenter les privilèges. La commutation de processus est réalisée par:
- JMP et CALL avec un descripteur de contexte de processus (TSS), ou un guichet de processus (TASK);
- une interruption avec un guichet de processus;
- par IRET avec le bit NT à un.

Les guichets apportent une indirection et une protection supplémentaires en restreignant l'appelant à certains points d'entrée: rôle du champ DEPLACEMENT dans la figure 14i.

On notera une certaine redondance dans l'iAPX386 entre les mécanismes mis en oeuvre aux niveaux page et segment, mais il convient de se rappeler que la pagination ou la segmentation peut être inhibée.

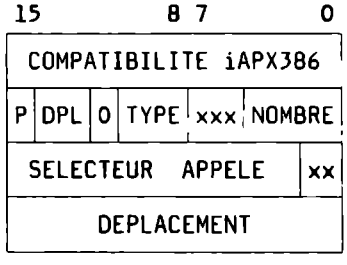


Figure 14j: format d'un descripteur de guichet (gate descriptor) dans l'iAPX286.

.de paramètres copiés de la pile de l'appelé s'il y a changement de privilège.
.aspect indirection du guichet (x: bit inutilisé)
.du point d'entrée dans le segment appelé

Multics utilise aussi les anneaux pour contrôler l'accès aux segments. Pour un segment de données il y aura deux limites:
- l'une définit le privilège minimum permettant l'accès en lecture,
- l'autre fixe le privilège minimum permettant l'écriture dans le segment.

Dans le cas d'un segment de code il peut y avoir augmentation ou diminution du privilège. La figure 14k donne un exemple des protections attachées à un segment de code et on y constate que:
- pour pouvoir exécuter le code appelé l'appelant doit avoir un anneau courant (nous utiliserons le terme de CPL, comme pour l'iAPX286) de numéro inférieur à une limite définie pour ce segment. Si CPL est égal à 7 l'appel sera rejeté;
- l'appelé a une fenêtre d'exécution (3 à 4 dans l'exemple): si le CPL de l'appelant se trouve à l'intérieur de cette fenêtre il n'y aura pas de changement d'anneau, pas de changement de privilège; situation analogue aux segments conformes (conforming, bit C=1) de l'iAPX286. Si le CPL de l'appelant est insuffisant (5 ou 6 dans l'exemple) il y aura accroissement de privilège et l'appelé

s'exécutera dans l'anneau de moindre privilège de sa fenêtre (4 ici) avec contrôle du point d'entrée correspondant. Si le CPL de l'appelant correspond à un privilège trop élevé (0 à 2 ici) il y aura diminution du privilège et l'appelé s'exécutera dans l'anneau 3 dans cet exemple, avec éventuellement copie préalable des paramètres qui résident dans des segments inaccessibles à l'appelé.

0	1	2	3	4	5	6	7	numéros des anneaux
LA	LA	LA	LAE	LAE	A	A		A: appelable E: exécutable L: lisible

Figure 14k: protections d'un segment de code sur Multics (DPS8M).

Sur **DPS7** comme sur iAPX la diminution de privilège lors d'un appel de code est interdite car ceci correspondrait à un code sécurisé (le SE par exemple) appelant du code non sûr. La structure des segments du DPS7 assure de plus que le segment n'est appelé qu'à travers des points d'entrée (entry point) bien répertoriés et définis en début de segment ("gate segment", GS=10 dans la figure 141). Un segment est protégé en lecture (champ RD dans la figure 141), en écriture (WR et WP), et en exécution (MAXR et EP). Pour que la lecture soit permise il faut que CPL soit inférieur ou égal à RD, pour l'écriture il doit être inférieur ou égal à WR mais encore faut-il que l'écriture soit permise (rôle de WP). Pour être exécuté il faut que EP soit positionné et que CPL ait une valeur convenable. Le lecteur aura noté que, contrairement à l'iAPX ou au DPS8, les descripteurs de segment du DPS7 ne sont pas accédés par le mécanisme normal d'adressage: ils ne se trouvent pas à l'intérieur d'un segment de descripteurs, mais dans des tables définies par d'autres règles d'accès.

0	1	2	3	4	5	6	7	8 ... 31
P	I	O	A	U	W	GS		BASE
RD		WR		MAXR		WP	EP	TAILLE

Figure 141: format d'un descripteur de segment direct (I=0) sur DPS7.

Sur **MV** de Data General les notions de segment et anneau sont confondues, toute donnée dans l'anneau j est accessible aux CPL inférieurs ou égaux à j. Comme dans le DPS7 il ne peut y avoir qu'augmentation ou conservation de privilège lors d'un appel, c'est-à-dire continuation dans l'anneau CPL ou passage à un anneau de numéro inférieur. Dans ce second cas l'appel doit passer par un guichet (gate) qui assure que l'exécution de l'appelé commence bien en début de segment. De plus l'appel peut être réalisé par numéro, l'édition de liens (linkage) étant faite par le guichet.

Le **VAX** dispose aussi d'un système proche des anneaux (modes d'accès dans la terminologie DEC) et l'appel de procédure ne peut s'y faire qu'en conservant ou accroissant les privilèges: rôle des ins-

tructions Change Mode (CHMK, CHME, CHMS, CHMU) qui provoquent une interruption analogue à celle induite par la SVC d'IBM. Dans\xx+6e mode d'accès K (Kernel, noyau) s'exécutent le videur (swapper), le responsable de la pagination (pager), le gestionnaire des interruptions et des exceptions. Dans le mode d'accès E (Executive) on trouve RMS, les méthodes d'accès. Le mode d'accès S (Supervisor) comporte l'interpréteur de commandes CLI.

14.6.5 – Listes d'accès et capacités.

Dans l'introduction de ce chapitre nous avons reproduit un modèle théorique de représentation des droits, la matrice des autorisations, et nous avions averti le lecteur que les implémentations réelles étaient différentes. Elles sont de deux types: les listes d'accès (ACL) et les capacités.

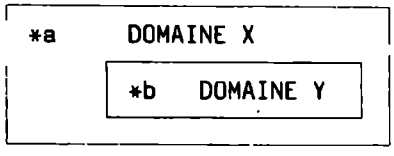
Les **listes d'accès** sont plus compactes que la matrice puisque qu'elles ne contiennent que les cases non vides de la matrice des autorisations. Elles relèvent d'un choix orienté **objet** puisque chaque liste, chaque colonne de la matrice, est associée à un objet. Les listes d'accès sont efficaces pour gérer les droits d'accès à un objet car elles constituent en fait un regroupement des droits au niveau de l'objet, puisqu'il y a regroupement à son niveau.

Un choix orienté **sujet** conduit à ce que l'on appelle des **capacités** (capacity, capability): à chaque sujet est associée une liste de ses droits, ce qui revient à regrouper les cases non vides d'une ligne de la matrice des autorisations. Chaque capacité est une sorte de descripteur protégé d'un objet associé, un ticket permettant d'avoir un certain droit d'accès à l'objet spécifié. La présence du descripteur montre que l'accès à l'objet est toujours indirect; il peut même y avoir passage par un processus gérant l'objet. Le contrôle joue ainsi au niveau de la requête et non pas au niveau de l'objet. Entre liste d'accès et capacité il y a donc une différence analogue à celle entre barrage et robinet à la source. Les capacités sont jugées plus efficaces lors de l'exécution car plus susceptibles de résider en mémoire, avec le sujet (un processus) qui les possède. Elles sont aussi jugées plus efficaces pour le transfert des droits entre sujets.

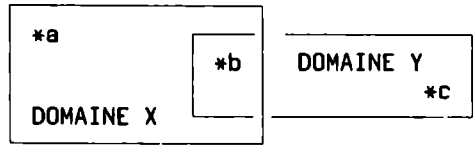
Si les listes d'accès sont fréquemment utilisées pour les objets du type fichier, les capacités sont plutôt utilisées pour les objets de type segment. Une table de descripteurs de segments, non modifiable par l'utilisateur, est proche d'une liste de capacités puisqu'elle définit pour un processus les segments et les modes d'accès auxquels il peut prétendre. Une liste d'accès est d'un emploi plus aisé pour l'utilisateur, mais sur un gros système elle peut conduire à de longues recherches et ne visualise pas facilement le domaine accessible à un utilisateur donné. Les ACL se prètent aussi plus facilement à la suppression des droits. Notons que le caractère incontrôlé du vocabulaire informatique conduit certains auteurs ([JON75]) à utiliser capacité dans le sens d'ACL.

Le **domaine** est une notion typique des systèmes à capacité. Le domaine courant d'un sujet est l'ensemble des objets auxquels il peut accéder. Un exemple de domaine est le mode de fonctionnement: dans les

machines classiques il y a deux modes, privilégié et non privilégié; dans les machines à anneau il y aura au moins 4 domaines, mais hiérarchisés: pour qu'un objet a soit accessible dans un domaine X, mais pas dans un domaine Y, il faut que X soit hiérarchiquement supérieur à Y, et tout objet b accessible à Y est aussi accessible à X. Un objet tel que c dans le système à capacité ne peut pas exister dans un système hiérarchique:

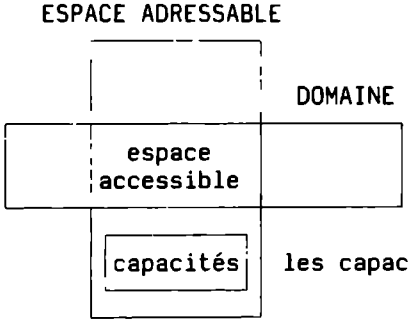


SYSTEME HIERARCHIQUE



SYSTEME A CAPACITES

La table des segments définit l'espace adressable d'un processus, espace plus ou moins ouvert selon l'anneau considéré dans un système hiérarchique. Dans un système à capacités, la notion de domaine ne coïncide plus avec celle d'espace adressable du processus, il n'y a pas de table de segments et la liste des capacités qui en est proche peut varier au cours de l'exécution du processus. Le sujet est le couple [processus, domaine], un processus qui change de domaine change de capacités. Cette association dynamique entre processus et domaines permet de ne donner que le minimum de droits nécessaires à un instant donné, il n'y a plus besoin de donner en début d'exécution l'ensemble des droits auxquels le processus fera appel tout au long de son exécution.



Un des aspects essentiels des domaines est le découpage de l'espace adressable en deux parties:
- le code et les données d'une part,
- les capacités d'autre part,

d'où le nom de "mémoire partitionnée" parfois utilisé ([SHA77]).

les capacités ne sont modifiables que par le SE.

Ainsi dans GCOS8 on ne retrouve pas le mécanisme des anneaux, mais trois modes de fonctionnement: mode maître privilégié (privileged master), mode maître, mode esclave (non privilégié). Un processus a un espace adressable défini par son numéro de WSP (Working SPace) courant et son domaine courant est inclus dans cet espace adressable, restreint par ses capacités rangées dans le segment des liens (linkage segment LS). L'instruction CLIMB permet de passer d'un domaine à un autre avec éventuellement changement de WSP. L'instruction SHRINK permet de restreindre la zone accessible par un descripteur.

D'autres systèmes commercialisés et basés sur les capacités sont le système 250 de Plessey (1974), le système 38 (S/38, 1981) d'IBM avec ses "pointeurs système" ([BER78]), l'iAPX432 dont on a beaucoup parlé, mais que l'on a peu vu.

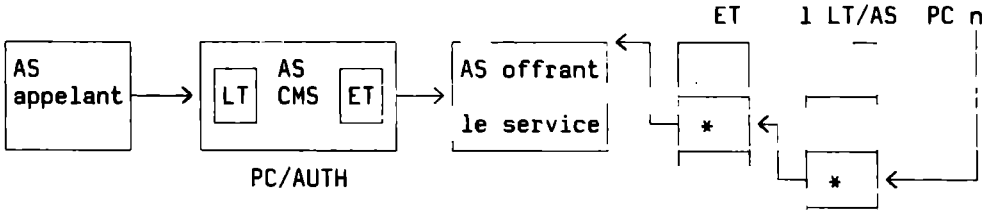
Les privilèges de VMS correspondent à des capacités restreintes à quelques objets. Utilisés en conjonction avec les ACL, ils sont analysés en dernier lors de chaque accès.

Les services "inter espaces" (CMS, Cross Memory Services) de MVS présentent aussi un aspect "capacitif", ils constituent le support logiciel des fonctions DAS présentées au chapitre 9 et permettent aux utilisateurs "autorisés" (utilisateurs en mode privilégié ou avec les clés 0 à 7) d'appeler du code ou d'accéder à des données qui se trouvent dans d'autres AS. Un AS particulier appelé PC/AUTH contient le code et les blocs de contrôle de CMS.

CMS: logiciel
DAS: matériel

Les AS offrant des services, c'est-à-dire pouvant être appelés ou accédés depuis d'autres AS sont inscrits dans une table ET (Entry Table). Chaque entrée de ET identifie l'AS serveur et les AS autorisés à l'appeler. Des tables LT (Linkage Table) représentent l'aspect capacitif de CMS: chacune de ces tables définit les AS serveurs auxquels l'AS associé a droit.

L'appel se fait par l'instruction PC avec comme opérande un numéro n identifiant le service, numéro unique dans MVS; cette instruction utilise les deux tables ET et LT pour accéder à l'AS serveur si l'accès est autorisé (cf. [MAR86]):



14.6.6 – Utilisation des piles.

De nombreux processeurs ont des piles comme nous l'avons souligné dans le tome I, piles pour le passage des arguments lors de l'appel des procédures, utilisées aussi pour stocker les variables locales d'une procédure. Le passage des arguments soulève un problème de protection puisque des morceaux de code de privilèges différents peuvent s'appeler.

Un processeur simple comme celui du 6502, abondamment utilisé dans le tome I, dispose d'une seule pile, à l'instar des microprocesseurs 8 bits ou 16 bits comme le 8086. Des microprocesseurs 16 bits comme le 68000 ont déjà deux piles, une pour le mode privilégié, une autre pour le mode non privilégié. L'iAPX286 a 4 piles par processus, une par anneau. Le VAX a 4 piles par processus aussi, une par mode d'accès, plus une cinquième pile, globale, utilisée pour les interruptions.

La multiplicité des piles est justifiée par les raisons suivantes:
- chaque processus doit avoir un espace adressable séparé;
- les programmes des utilisateurs ne doivent pas dépendre de données changeantes en fin de pile. Ceci peut se produire si le programme utilisateur appelle une fonction système qui viendra ranger ses variables dans la pile, derrière celles de l'appelant. Avec une pile par anneau, le système qui s'exécute dans des anneaux plus privilégiés rangera ses variables dans une autre pile;
- les programmes privilégiés ne risquent pas de voir leur pointeur de pile détruit par un programme moins privilégié défectueux;
- les programmes privilégiés ne dépendent pas de la taille de la pile de leur appelant moins privilégié;
- la plus grande partie des piles peut être paginée.

14.7 – Sur mémoire périphérique.

La protection de l'accès aux mémoires périphériques est réalisée par des moyens physiques, que nous ne ferons que lister, et par des mécanismes logiques construits par le système d'exploitation.

Les moyens physiques sont les boutons que l'on trouve sur les unités (dérouleurs de bandes, unités de disque), les dispositifs associés aux volumes: anneau pour les bandes, languette pour les cassettes, encoche dans le cas des disquettes, etc. Ces protections physiques peuvent être simulées par des commandes du système d'exploitation comme la commande SET de RT-11 de DEC qui permet de protéger un disque contre toute écriture.

Les protections logiques sont basées sur l'accès restreint aux instructions d'entrée/sortie et l'identification de l'utilisateur au moment de sa connexion au système. Nous trouverons ainsi plusieurs dispositifs, certains faisant l'objet d'un simple rappel puisque traités dans d'autres chapitres.
- **étiquettes de volume**: elles permettent de s'assurer que le volume est bien celui auquel on pense accéder;
- **mots de passe**: associés au volume ou au fichier ils en réservent l'accès aux utilisateurs les connaissant;
- **listes d'accès**: les noms des utilisateurs pouvant accéder à la ressource figurent dans une liste associée à cette ressource;
- **alarmes**: quand l'objet ainsi protégé est accédé selon les modes d'accès surveillés, une alarme est générée. Dans VMS ce dispositif est mis en action au niveau des listes d'accès: avec la commande

```
SET FICH/ACL=(ALARM=SECURITY,ACCESS=READ+WRITE+DELETE+CONTROL+FAIL+SUCCESS
```

  tout accès (en lecture, écriture, etc.) réussi (SUCCESS) ou refusé (FAIL) au fichier FICH provoquera l'émission d'une alarme sur le poste de travail du responsable de la sécurité;
- **isolation des accès** (AIM, Access Isolation Mechanism dans Multics);
- **chiffrage**: le contenu du fichier est sujet à un cryptage/décryptage lors des accès;

- **réservation/libération** (reserve/release) de périphériques dans le cas des multisystèmes par exemple;
- effacement des zones après libération ou avant allocation des ressources;
- contrôle de l'espace accédé: la commande d'E/S (CCW, Channel Command Word) "set file mask" permet d'empêcher les accès en dehors de l'extension courante sur un disque relié à un 370 d'IBM.

Notons que:
- l'assimilation des volumes et périphériques à des fichiers permet de les faire bénéficier de tous les mécanismes de protection mis en place pour les fichiers;
- la mémoire adressable à un seul niveau (cas de Multics, du S/38) étend aux fichiers les mécanismes créés pour les segments.

14.8 – Privilèges, limites, quotas.

Le privilège est un droit associé au sujet mais portant sur un ensemble d'objets et/ou d'opérations. VMS associe ainsi un "masque des privilèges", mot de 64 bits tiré du fichier des autorisations (UAF), chaque bit positionné correspond à l'attribution du privilège correspondant.

Une limite est un droit sur une ressource, une quantité maximum d'une ressource ré-utilisable comme la mémoire par exemple.

Un quota est aussi un droit sur une ressource mais la ressource n'est pas réutilisable: le temps processeur en est un exemple.

14.9 – Autres protections.

Les protections se présentent sous de nombreux aspects; nous citerons encore:
- le contrôle de l'accès aux commandes en définissant des classes d'opérateur. VM distingue ainsi les classes suivantes:

A: opérateur système principal	E: analyste système
B: accès aux ressources système	F: représentant du service
C: programmeur système	G: utilisateur courant
D: opérateur du spoule	H: réservé à IBM

- l'APF (Authorized Program Facility) permet de supprimer certains contrôles (bypass) pour des programmes autorisés, comme la vérification du propriétaire lors de l'ouverture d'un fichier par exemple ([DUB86]);
- dans le monde MVS des logiciels comme RACF ou ACF2 doivent être ajoutés pour accroître les protections:
  * ACF2 est un système "fermé" dans lequel les objets non déclarés sont automatiquement protégés;
  * RACF d'IBM est dit "ouvert": les objets non déclarés n'ont aucune protection. Encore que l'option ADSP permette de protéger tout fichier ([DUB86]). Les objets sont protégés par un mécanisme d'ACL avec un découpage des utilisateurs en individus et groupes, un individu pouvant appartenir à plusieurs groupes.

Un autre moyen d'améliorer la protection est de lier les droits de l'utilisateur à son mode de connexion. Nous l'avons cité dans le cas

de GCOS6 dans le chapitre concernant la gestion des données, VMS distingue sept types de connexion (login): locale, commutée (dialup), distante (remote), réseau (network), lot (batch), détachée (detached process), processus fils (subprocess).

14.10 – Partage.

Nous avons traité ci-dessus des moyens d'interdire tel type d'accès à tel objet du simple point de vue statique de l'accès permis ou non. Le problème du partage intervient lorsque plusieurs sujets (processus) ayant le droit d'accéder à la ressource exercent simultanément ce droit. Pour certaines ressources il n'y a pas de limite au nombre d'accès simultanés: cas d'une zone mémoire. Pour d'autres cette limite est très précisément définie, elle est égale à un (cas d'un processeur, d'un dérouleur de bande) ou à quelques unités. Dans le cas de ressources pré-emptives comme le processeur aucun dispositif particulier n'est nécessaire en dehors de mesures comme la sauvegarde du contexte. Si la ressource n'est pas pré-emptive, il convient de mettre en place un réglement qui permettra d'assurer des accès cohérents, avec mise en attente, active ou non, des processus.

14.10.1 – Verrous et sémaphores.

Verrous et sémaphores sont des exemples de mécanismes régulant l'accès aux ressources qui supportent un nombre limité d'accès concurrents. Tous deux résolvent le problème de la ressource non partageable, c'est-à-dire avec un seul accès à la fois; le sémaphore, grâce à un compteur, permet de gérer l'accès aux ressources supportant plusieurs requérants simultanés.

Dans le cas de MVS un verrou (lock) est une étiquette sur laquelle le détenteur d'une ressource inscrit son identité, pour sérialiser les accès aux ressources non partageables mais réutilisables en série. Les verrous y sont d'ailleurs hiérarchisés afin d'éviter les interblocages: un processus ne peut obtenir que des verrous plus élevés que ceux qu'il possède déjà.

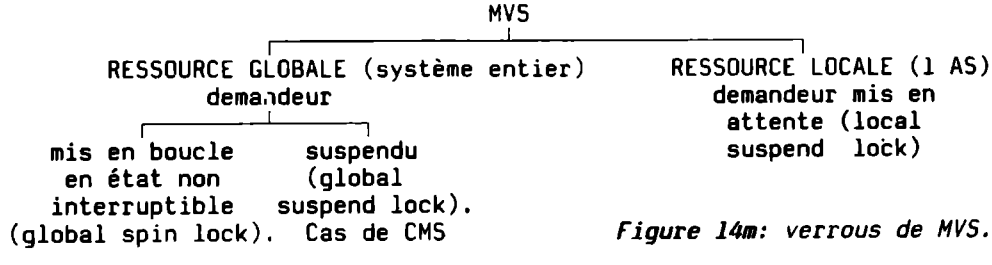


Figure 14m: verrous de MVS.

MVS distingue verrou local et verrou global, le verrou local étant utilisé pour la synchronisation à l'intérieur d'un espace d'adresses. On y trouve plusieurs verrous parmi lesquels:
- le verrou DISP du distributeur;
- le verrou CMS pour synchroniser les AS qui ont quelque chose à se

dire;
- les verrous du GESP (I/O Supervisor lock), un par type de bloc de contrôle: un par UCB, IOSCAT (canaux physiques disponibles), IOSLCH (un par file d'attente sur canal logique), IOSYNC;
- le verrou de pagination: un pour la mémoire centrale (SALLOC géré par RSM et VSM), et un pour la mémoire secondaire (ASM);
- les verrous de VTAM: TPNCB, TPDNCB, TPACBDEB, etc.

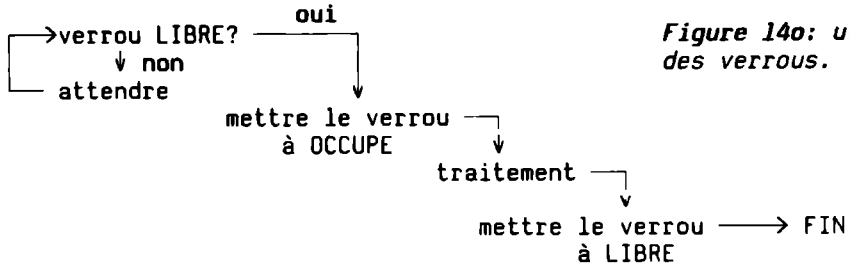


Figure 14o: utilisation des verrous.

Si les verrous vus plus haut sont principalement destinés à résoudre les problèmes posés par le multitraitement, d'autres services de verrouillage sont mis en oeuvre pour résoudre ceux posés par la multiprogrammation ([DUB86]): ils correspondent aux primitives QUEUE et DEQUEUE (mise en file d'attente) dans les OS MFT, MVT, MVS d'IBM. Cette technique est généralement utilisée pour les accès aux fichiers et aux volumes.

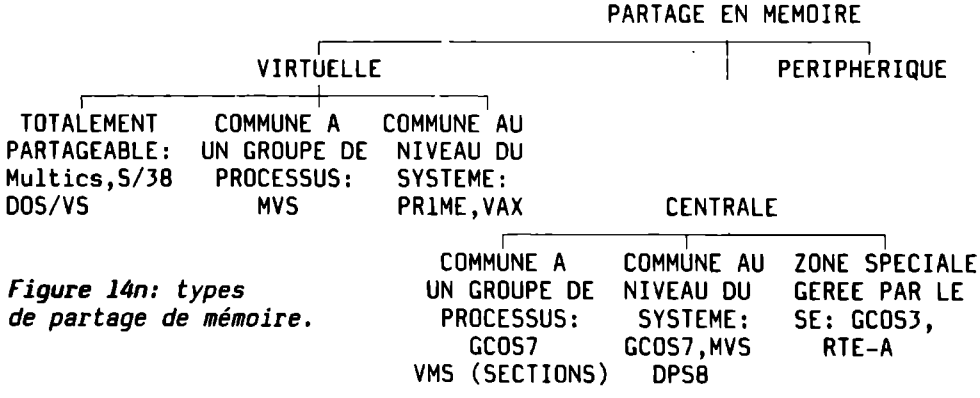


Figure 14n: types de partage de mémoire.

14.10.2 – Partage de données en mémoire.

Le partage de zones mémoire peut être réalisé par l'architecture matérielle de la machine: il en va ainsi sur l'iAPX286 où la table GDT des descripteurs globaux est commune à tous les processus; la moitié des segments est donc commune à tous les processus; il en va de même sur PRIME au niveau des segments, sur VAX au niveau des pages. Multics ou le S/38 sont dans un cas analogue, mais tout l'espace adressable peut y être partagé si les droits d'accès sont donnés à tous les

processus. Dans MVS par contre chaque groupe de processus (chaque étape) a son propre espace adressable appelé AS en mémoire virtuelle.

Ce partage créé par l'architecture du matériel peut être complété par des constructions du SE, le SE peut aussi le créer de toute pièce: ainsi dans les DPS7 chaque processus a son propre espace adressable en mémoire virtuelle, seule l'affectation de la même adresse de base, réelle donc, à plusieurs segments peut faire en sorte qu'ils correspondent à la même information. Le DPS7 offre d'ailleurs deux moyens d'associer, en mémoire réelle notons le, le même contenu à deux segments différents:

- en rangeant la même adresse réelle dans les deux descripteurs (figure 14o);

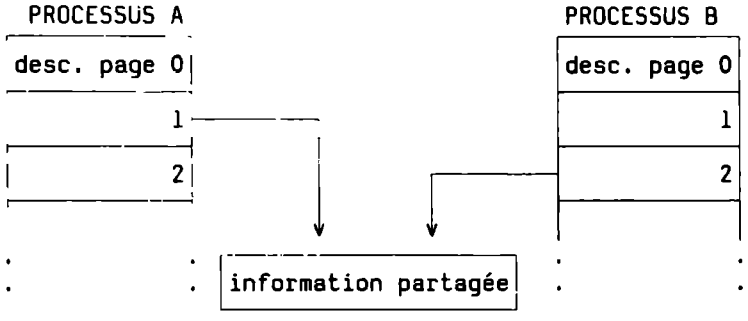


Figure 14o: partage par duplication de l'adresse dans les descripteurs de page ou de segment.

- en utilisant des **descripteurs indirects**: seul un descripteur contient l'adresse réelle du segment, les autres contiennent l'adresse réelle de ce descripteur (figure 14p).

La seconde solution est rarement utilisée par GCOS7 bien qu'elle ne rende qu'un seul descripteur dépendant de l'adresse réelle du segment.

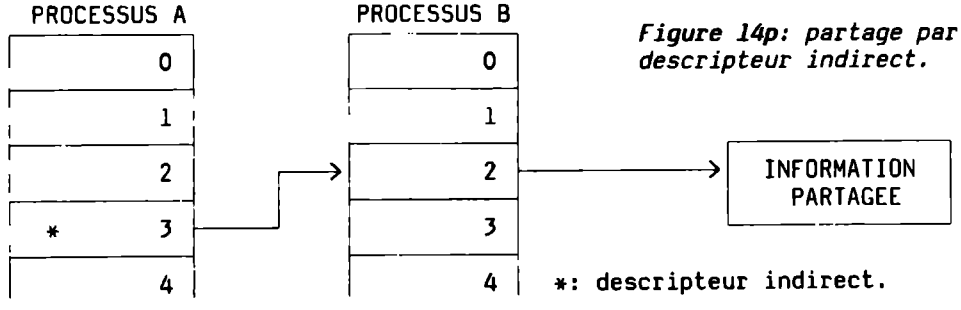


Figure 14p: partage par descripteur indirect.

VMS crée, en plus du partage en mémoire virtuelle offert par l'architecture matérielle du VAX, un partage en mémoire centrale analogue à celui de GCOS7: ce sont les **sections**.

chapitre 15

langages de commande

15.1 – Types de langage.

L'utilisateur d'un ordinateur dialogue à différents niveaux d'architecture comme le rappelle la figure 15b. Nous nous intéressons plus particulièrement dans ce tome au langage par lequel l'utilisateur dialogue avec le système d'exploitation, langage que l'on trouve sous deux formes:

- **non-interactive** dans la dimension du traitement par lots,
- **interactive** dans le cas des SE interactifs comme Unix, ou dans les sous-systèmes temps partagé des SE, comme TSS sous GCOS3, TSO sous MVS.

Dans le cas des langages interactifs, nous distinguerons deux types de langage adaptés à deux types d'utilisateur:

- le langage destiné à l'utilisateur du système d'exploitation, un programmeur par exemple, langage qui porte le nom générique de JCL (Job Control Language),
- le langage destiné au **pupitreur** et génériquement appelé OCL (Operator Control Language).

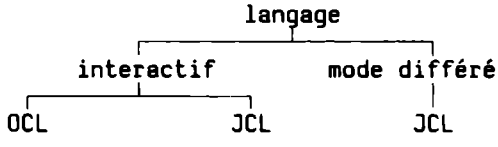


Figure 15a: typologie des langages de commande du système d'exploitation.

15.2 – Message d'invitation.

Dans le cas du mode différé, l'utilisateur indique que la ligne (la carte) correspond à une commande du SE en utilisant comme début de commande une séquence imposée par le SE:

- le caractère **$** dans les GCOS, dans VMS,
- les caractères // sous les OS et DOS d'IBM,
- le caractère : sur Tandem,
- le caractère **@** sous OS1100, etc.

Pour les langages de commande interactifs, l'utilisateur ne frappe que la commande, le SE sachant toujours reconnaître ce qui est une

commande. Les commandes sont généralement frappées en réponse à une invitation du SE, invitation qui se matérialise par quelques caractères affichés que les américains appellent un "**prompt**":

- ce sera le nom de l'unité de disquette (ou de disque) courante suivi du signe "plus grand que" sous MS-DOS:

  A>

  par exemple si la disquette A est la disquette courante,
- VMS affichera un dollar "$" comme message d'invitation,
- RTE-A affichera "VCP>" pendant le démarrage du SE, "CI>" une fois RTE-A lancé,
- Multics affichera "M→" sur la console opérateur lorque Multics aura été lancé, il affichera "BOS→" pendant la phase d'amorçage, etc.

Ce message d'invitation peut se révéler très utile pour les néophytes car il indique avec qui l'utilisateur dialogue. Les applications, les utilitaires auront leurs propres messages d'invitation, éventuellement plusieurs si nécessaire. Ainsi les éditeurs de texte en mode ligne afficheront (cas d'EDIT de VMS) des invitations différentes selon que l'on est en mode commande ou en mode entrée (input).

L'utilisateur averti pourra par contre supprimer tout message d'invitation (commande "RDY -OFF" sous Primos) afin de n'avoir que les échanges utiles. Il pourra aussi supprimer les indications de temps processeur, de consommations de ressource que certains systèmes affichent après exécution de chaque commande (Multics, Primos, CMS). Il pourra aussi personnaliser son environnement en définissant son propre message d'invitation qui remplacera celui offert par défaut par le système (commande PROMPT de MS-DOS).

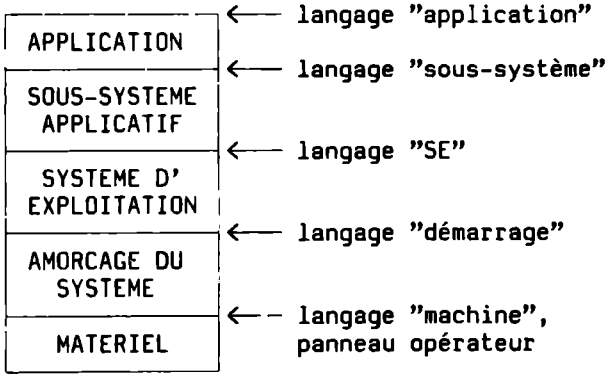


Figure 15b: les différents niveaux de dialogue entre l'utilisateur et l'ordinateur.

15.3 – Syntaxe des langages.

Les langages de commande peuvent être classés en plusieurs catégories:

- orientés **verbe**, ils produisent alors un grand nombre de commandes différentes. Les GCOS, OS1100, MCP de Burroughs, DOS d'IBM, les langages interactifs font partie de cette catégorie;
- orientés **paramètres** ils disposent d'un nombre restreint de commandes avec de nombreux paramètres. Le JCL de MVS fait partie de cette catégorie. On distingue encore deux types de paramètre:

* **positionnels**: leur position dans la commande indique sur quel dispositif ils portent,
* à **mot-clé**: la valeur est précédée d'un mot clé qui indique de quel dispositif il s'agit. Ils peuvent, contrairement aux paramètres positionnels, être placés dans n'importe quel ordre. Le JCL de MVS entre dans cette catégorie pour les paramètres de premier niveau, il comporte ainsi trois commandes principales:
  . // JOB qui définit un nouveau travail,
  . // EXEC qui définit le début le début d'une nouvelle étape,
  . // DD qui définit une ressource périphérique utilisée par l'étape;

- dans le cas des systèmes interactifs on a aussi les langages orientés **question/réponse**;
- toujours dans le cas des systèmes interactifs, les langages orientés **menus**,
- les langages basés sur les **icônes**. Le MacIntosh d'Apple en offre un exemple.

La frappe de commandes peut faire l'objet de simplifications comme:
- les **abréviations** imposées, c'est-à-dire une forme abrégée bien définie pour chaque commande. Primos dispose de ce type d'abréviations, Multics aussi (short form);
- les abréviations par **radical discriminant** où l'utilisateur doit frapper les premières lettres de la commande. Le nombre de lettres frappées doit être tel qu'il n'y a pas d'ambiguïté pour le SE. VMS est dans ce cas: la commande DIRECTORY pourra être frappée sous les formes DIRECTOR, DIRECTO, ... DIR, mais pas DI car alors le système ne sait pas si l'utilisateur a voulu frapper DIRECTORY ou DIFFERENCES.

Certaines interfaces homme/machine permettent les trois modes: nom complet, abréviation mnémonique imposée, radical discriminant.

Un autre système d'abréviations permet à l'utilisateur de **personnaliser**, de redéfinir le nom des commandes, d'en combiner plusieurs pour simplifier la frappe de commandes fréquentes. Ceci correspond à ce que l'on appelle des **processeurs d'abréviations**. Primos par exemple offre cette fonctionnalité et l'utilisateur passe en mode abréviation en frapppant la commande "AB -ON", il pourra revenir au mode normal par "AB -OFF". La commande "AB -LIST" lui permettra de lister tout ou partie du fichier d'abréviations actif, "AB -EX" permettra d'exécuter une commande sans passer par le processeur d'abréviations, etc. VMS offre quelque chose d'équivalent avec la gestion des "noms logiques", noms définis par la commande $DEFINE. La partie du SE qui traite les commandes s'appelle:
- "processeur de commandes" (command processor) très fréquemment,
- COMMAND.COM sur MS-DOS,
- Shell sur Unix,
- "listener" sur Multics, GCOS6,
- CCP (Console Command Processor) sur CP/M,
- CLI (Command Line Interpreter) sur MP/M, RSX-11M, COMINT sur Tandem,
- CI sur RTE-A, etc.

Certains de ces langages (DCL de VMS, EXEC de CMS, Shell d'Unix) cons-

tituent de véritables langages de programmation avec tests, branchements, boucles, variables.

Les langages de commande du mode différé sont généralement moins développés mais beaucoup permettent l'exécution **conditionnelle** des étapes. Ainsi le paramètre COND des commandes // EXEC des OS d'IBM permet d'exécuter l'étape correspondante en fonction du **code condition** positionné par l'étape précédemment exécutée; il correspond au "IF ERRORLEVEL" de MS-DOS; il ne permet cependant pas de faire des sauts en arrière. Les commandes $ IF, $ WHEN, $ label., $ GOTO, $ BREAK de GCOS3 offrent déjà plus de souplesse, le "code condition" y étant remplacé par le "mot de commutation" (switch word).

Les **macrocommandes** sont offertes par certains langages (commandes // PROC et // PEND des OS d'IBM) en dehors du mécanisme d'abréviations vu plus haut. La commande $ SELECT de GCOS3 offre quelque chose d'analogue, cependant plus proche du **fichier de commandes** classique des systèmes interactifs. Sous GCOS7 on aura $ INVOKE. Le passage de **paramètres** est très souvent offert: %i (i=0,1,2,etc.) sous MS-DOS, EXEC2 de CMS, &nom des OS d'IBM, $i de RTE-A par exemple. La commande $ PARAM offre une fonctionnalité analogue sous GCOS3. L'exemple ci-dessous, tiré de [LAN85] donne deux versions des mêmes commandes, L'une lancée dans le mode lot, l'autre en temps partagé.

```
//          EXEC PGM=COPY,PARM='COPY ALL'
//STEPLIB   DD   DSN=NAME.LINKLIB,DISP=SHR
//FROM      DD   DSN=NAME.OLD.FILE,DISP=SHR
//TO        DD   DSN=NAME.NEW.FILE,DISP=OLD
//SYSPRINT  DD   SYSOUT=*
```

JCL sous MVS copiant le fichier NAME.OLD.FILE dans le fichier de nom NAME.NEW.FILE. Pour le programme COPY, FROM est le nom du fichier à copier.

```
alloc dd(from) dsn('name.old.file')
alloc dd(to)   dsn('name.new.file') old
alloc dd(sysprint) ds (*)
call  'name.linklib(copy)' 'copy all'
```

La même copie mais lancée sous TSO. On notera que les commandes sont frappées dans l'ordre inverse: CALL qui correspond à EXEC vient en dernier.

15.4 – Procédures cataloguées.

Il est fréquemment possible de stocker sur disque des ensembles de commande souvent utilisées. IBM parlera de procédures cataloguées (généralement dans le fichier SYS1.PROCLIB sous MVS), les autres de fichiers de commande: fichier de type EXEC sous CMS, .BAT sous MS-DOS, .CMD sous VMS, sous RTE-A, .EC sous Multics, etc. Nous avons vu dans le chapitre 4 comment lancer ces fichiers de commande.

L'appel de procédures ou de fichiers de commande depuis des procédures ou fichiers de commandes (imbrication, nesting) n'est pas toujours possible. MVS ne le permet pas. Notons que les langages à base d'icônes ne se prêtent pas facilement aux fichiers de commandes.

15.5 - Règles de recherche.

Lorsque l'utilisateur frappe une commande le SE doit d'abord trouver la commande. Ainsi sous MS-DOS il examinera successivement:

- les commandes contenus dans COMMAND.COM et donc résidentes en mémoire centrale,
- le répertoire courant de l'utilisateur, ce qui permet d'avoir plusieurs commandes de même nom, mais différentes selon les utilisateurs, selon les versions,
- le répertoire défini par la commande PATH.

Ces règles de recherche (search rules) sont, selon les SE, imposées par le SE, ou modifiables par l'utilisateur. Unix cherchera ainsi successivement la commande dans le repertoire courant, dans le repertoire "/bin" où l'on place les commandes les plus courantes, puis dans "/usr/bin"; les règles de recherche y sont modifiables par la variable PATH du Shell.

chapitre 16

génération, installation, démarrage, configuration

16.1 – Démarrage du système.

Le système, constitué par l'ordinateur et son système d'exploitation, ne fonctionne généralement pas de façon ininterrompue. Il connaît des arrêts volontaires ou involontaires (dus à des pannes par exemple). Lorsque le système est remis en service on dira qu'on le **démarre**. Il existe deux types de démarrages (boot):

- à **froid** (cold boot): le système repart alors totalement à partir de zéro et il subit une initialisation complète depuis un état initial, toujours le même. Il recrée complètement l'environnement d'exploitation;
- à **chaud** (warm boot): le système repart depuis un état intermédiaire, dans le but d'assurer une relance plus rapide, et en perdant le minimum de travail déjà accompli. Il repart non pas de l'état initial zéro qui correspond au démarrage à froid, mais à partir du dernier état cohérent avant l'arrêt; il suppose que certaines informations du "boot" précédent sont utilisables.

Le démarrage à froid de MS-DOS correspond à la mise sous tension du micro-ordinateur (ou à l'appui simultané sur les touches CONTROL, ALT et DELETE, encore que cet appui soit parfois inopérant). Cette action est reçue sous forme d'interruption (cf. tome III) par le processeur, interruption appelée RESET dans les 8088. Elle y dispose d'un vecteur d'interruption ou séquence de code qui se trouve à l'adresse FFFF0 en mémoire centrale. On trouve généralement à cette adresse une instruction de branchement vers la séquence de démarrage (on trouvera les termes de BOOT, IPL dans les brochures des fournisseurs ou fabricants) qui se trouve en mémoire morte.

Cette séquence en mémoire morte, immuable a priori pour un micro-ordinateur donné, a pour rôle de charger en mémoire vive une nouvelle séquence de démarrage plus complexe et plus évolutive qui se trouve sur le secteur zéro de la disquette A ou du disque C. La séquence de démarrage en mémoire morte se termine par un saut vers la zone (d'adresse 7C000 en hexadécimal) où a été rangé le contenu du secteur zéro du périphérique de démarrage. Cette séquence charge à son tour le

SE, à savoir MS-DOS qui se trouve stocké sur le disque (ou la disquette) de démarrage sous forme de deux fichiers "cachés" (hidden dans la terminologie MS-DOS), IO.SYS (chargé à l'adresse 600) et MSDOS.SYS (noms MS-DOS de deux fichiers appelés IBMIO.COM et IBMDOS.COM dans PC-DOS). IO.SYS prend la main, lit le fichier de configuration CONFIG.SYS, puis donne la main à MSDOS.SYS qui charge COMMAND.COM et le lance. La partie d'initialisation de COMMAND.COM exécute le fichier AUTOEXEC.BAT s'il existe puis passe le contrôle à la partie semi-permanente de COMMAND.COM qui correspond à l'interpréteur de commandes. Le démarrage est alors terminé, le message d'invitation du système (prompt) s'affiche sur l'écran et l'utilisateur peut dialoguer avec le SE.

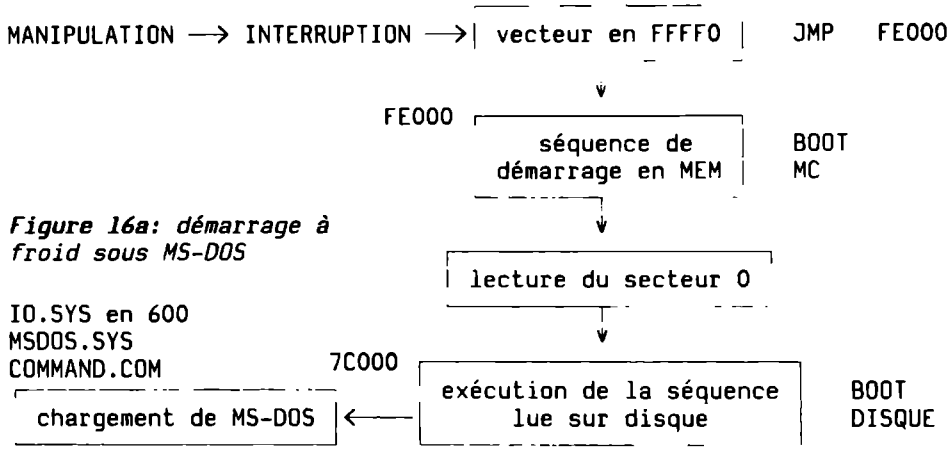


Figure 16a: démarrage à froid sous MS-DOS

Sur un gros SE les choses sont un peu plus complexes et [IBM80C] décomposera le démarrage à froid en plusieurs étapes illustrées par la figure 16b.

Sur les SE des minis, moyens ou gros ordinateurs, le démarrage fait généralement intervenir un SE simplifié, parfois une version précédente, mono-utilisateur, du SE: ainsi BOS (Boot OS) pour Multics, Primos II pour Primos, BOOTEX sur RTE-A, SCP pour GCOS7.

Durant ce démarrage il y a un dialogue qui se déroule entre le pupitreur et les routines de démarrage. Dans le cas de MVS, le fichier contenant les routines de démarrage réside sur un volume appelé SYSRES qui contient un certain nombre de fichiers appelés:

- SYS1.NUCLEUS, le noyau résident, ainsi que les modules initialisant ce noyau (NIP, Nucleus Initialization Procedure Modules), les ressources (RIM, Resource Initialization Modules) et le planificateur maître. Ce noyau est chargé par l'IPL qui, lui-même, se trouve sur SYSRES;
- SYS1.LOGREC pour stocker les messages d'erreur intervenant durant l'exploitation (System Operation);
- SYS1.SVCLIB qui contient des routines du superviseur n'appartenant pas au noyau résident (SYS1.NUCLEUS), mais utilisées par NIP.

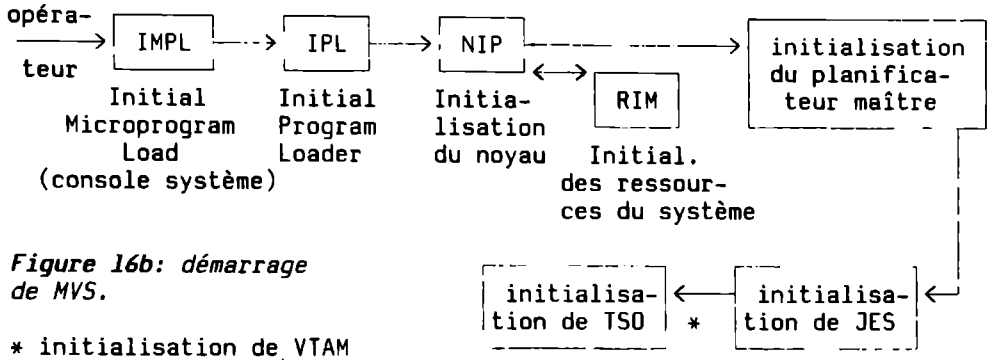


Figure 16b: démarrage de MVS.

* initialisation de VTAM

Le périphérique de démarrage est soit imposé (cas de MS-DOS qui essaie l'unité A en premier, puis le disque C), soit défini par le pupitreur au moyen d'un dispositif/dialogue lui permettant de définir l'unité de démarrage. Dans le passé ces unités de démarrage furent des lecteurs de cartes, des lecteurs de bande (d'où le terme de TOS, Tape Operating System). Dans le cas des machines microprogrammées (cf. tome I), il peut y avoir au départ un chargement des microprogrammes (phase IMPL de MVS dans laquelle sont chargés les microprogrammes de la console).

A l'occasion de cette phase de démarrage nous observons que:

- le démarrage du SE se fait de façon **téléscopique** (bootstrap); un algorithme minimal charge une séquence plus complexe qui, à son tour, charge du logiciel plus complet, etc., jusqu'à arriver au chargement du SE lui-même;
- le système d'exploitation contient du logiciel qui n'a qu'un rôle intérimaire, limité dans le temps, comme le démarreur d'un véhicule. Nous l'appellerons logiciel d'amorce.

Au niveau du vocabulaire, on trouvera encore les termes de "system startup", initialisation.

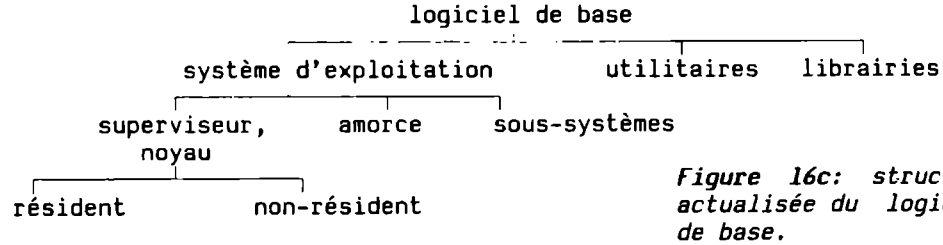


Figure 16c: structure actualisée du logiciel de base.

16.2 – Installation.

Nous venons de voir le démarrage du SE. Mais pour démarrer faut-il encore qu'il existe physiquement, qu'il soit **installé** sur l'ordinateur. Insérer un logiciel, et ici plus particulièrement le SE, fait généralement l'objet d'un certain nombre de manipulations regrou-

pées dans ce que l'on appelle l'installation d'un logiciel.

Un logiciel installé réside généralement sur disque et est alors aisément accessible pour les utilisateurs agréés. Mais le fournisseur ne livrera généralement pas un logiciel sur disque directement accessible:

- le disque n'est pas un support de portage des logiciels, il est encombrant, peu standardisé et de nombreux disques sont aujourd'hui inamovibles;
- le logiciel doit tenir compte de l'environnement du système informatique dans lequel il est inséré, il doit être **personnalisé** (customized), paramétré en fonction de cet environnement, des objets, des modes d'utilisation que les utilisateurs désirent.

Les utilisateurs de PC sont habitués à l'installation des logiciels de traitement de texte, des logiciels intégrés, etc., installation fréquemment réalisée en lançant un fichier de commande appelé INSTALL.BAT. L'installation prendra par exemple en compte le type du clavier (Azerty, Qwerty), le type de l'imprimante, les caractéristiques graphiques de l'écran (étendues, carte Hercules, etc.).

L'installation correspond donc à l'insertion physique d'un logiciel dans un système informatique et à son adaptation aux caractéristiques de l'environnement. MVS a été décomposé en SU (Selectable Unit), chaque SU est modifiable, installable séparément, ce qui facilite l'installation du SE ainsi que sa distribution. Si les SU ont disparu, le concept associé existe toujours. Pour un SE cette adaptation se traduira par la prise en compte:

- du **matériel**: la taille mémoire, le nombre et le type des périphériques, le modèle du processeur;
- des **utilisateurs**: nombre, niveaux de service, etc.

Face à ce problème d'adaptation, nous distinguerons deux types de SE:

- ceux qui ont besoin d'une **génération** (cas des OS d'IBM);
- ceux qui sont seulement paramétrés par l'utilisateur et ne font pas l'objet d'une génération (cas de GCOS3, Multics, Primos, MS-DOS, etc.). Dans le cas de MS-DOS ce paramétrage correspond au fichier CONFIG.SYS qui est lu à chaque démarrage. Dans le cas de Multics, le fichier system_start_up.ec jouera un rôle analogue. Dans Primos ce sera le fichier C_PRMO du répertoire C_MDNCO.

La dernière version de VSE/SP est annoncée comme étant prégénérée, c'est-à-dire prête à être installée et utilisée.

16.3 – Génération du SE.

La génération est un processus de sélection de modules, d'options, de paramètres ([IBM80G]) qui permet d'adapter une installation du SE à partir des logiciels fournis par le constructeur (DLIB, Distribution LIBraries chez IBM). Elle peut durer plusieurs heures et nécessiter un travail important et complexe. A tel point que les fournisseurs se sont sentis obligés de fournir des produits facilitant la génération du SE: IPO (Installation Productivity Option chez IBM pour MVS) est une nouvelle techique de livraison, installation et mise en service des systèmes; elle correspond à une version de logiciel testée avant livraison.

Pratiquement, la génération conduit à exécuter un certain nombre de travaux (jobs) en utilisant une version précédente du SE.

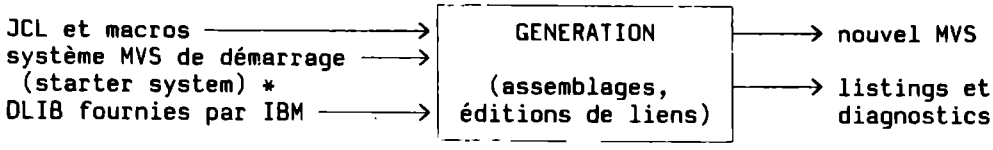


Figure 16c: environnement d'une génération de MVS
*(DLIB: Distribution LIBrairies, *: première génération seulement).*

Sur les systèmes orientés génération (qui sont à générer) la charge peut être importante. [IBM80C] indique pour une installation typique de MVS: deux personnes MVS avec une expérience JES, une personne pour TSO avec une expérience TCAM/VTAM, une personne pour IMS ou CICS.

16.4 – Personnalisation du SE.

D'une façon générale, le SE est personnalisé (modifié selon les besoins de l'utilisateur) à différentes étapes:

- lors de sa **génération** éventuelle;
- lors de chaque **démarrage** (bot, IPL, start-up);
- durant son **utilisation**, de façon statique, par la procédure de **patch** qui permet de modifier et/ou corriger le SE dans sa forme exécutable (le source du logiciel est de moins en moins fourni aux utilisateurs). Cette procédure de patch peut être très structurée et conçue pour éviter les erreurs de manipulation (en obligeant par exemple à préciser le contenu précédent de la zone modifiée). Un patch peut être mis dans le SE de façon:
  * statique, à l'intérieur des modules eux-mêmes;
  * dynamique, lors du chargement du SE, une des composantes de l'amorce lit le fichier des patches et les insère dans les différents modules concernés. Cette méthode permet une meilleur gestion des modifications et assure tout particulièrement une vérification aisée du "niveau" de patch. Niveau qui vient s'ajouter aux numéros de version de mise à jour pour qualifier précisément l'état d'un logiciel.

  Dans le cas d'IBM, les patches (PTF pour Program Temporary Fix) doivent être appliqués avant la génération;
- durant son utilisation de façon dynamique, en modifiant la copie du SE en cours d'exécution, et non pas la copie exécutable sur disque comme précédemment. Ces modifications sont faites par des commandes passées par le pupitreur (OCL) et conduisent aussi à des patches. Elles sont bien évidemment plus risquées dans la mesure où un patch erroné est susceptible de provoquer un mauvais fonctionnement du système. D'une façon générale, toute modification du SE doit faire l'objet d'un plan avec, avant la mise en exploitation, des tests de vérification de bon fonctionnement.

Certains SE permettent la mise à jour ou l'ajout sélectif des modules du logiciel qui le composent: les modules du SE peuvent être remplacés un par un.

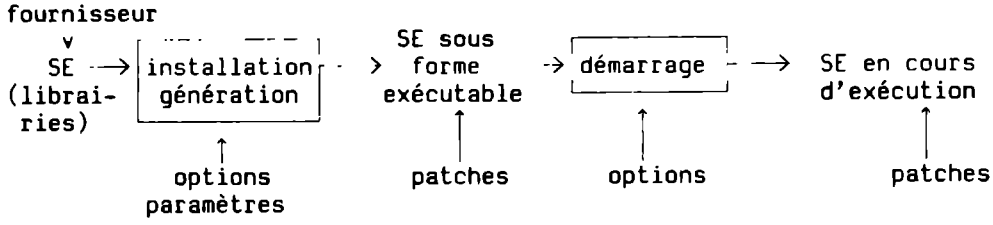


Figure 16d: les différentes étapes de l'adaptation du SE.

16.5 – Exits.

Les systèmes d'exploitation présentent un certain nombre de points où l'utilisateur, l'équipe système plus précisément, peut inclure des séquences de code qui lui sont propres et activées:
- lors de la connexion (login): cas du "login externe" de Primos, exits LOGON/LOGOFF des applications de VTAM;
- lors de la réception d'un message (status) d'erreur sur une liaison distante: cas de VTAM, des sous-systèmes sous MVS;
- pour accroître la protection, etc.

Ces points, répertoriés et documentés, sont appelés "exits" chez IBM.

16.6 – Configuration.

Le terme de "configuration", associé au matériel, comprend trois aspects:
- le choix des composants matériels ainsi que leur interconnexion au moyen de câbles,
- le positionnement de commutateurs, boutons, sur les équipements,
- une description du matériel indiquant au logiciel quel matériel est disponible et comment il doit être utilisé.

La reconfiguration **dynamique** correspond à la capacité du SE à accepter l'addition ou la suppression des équipements principaux comme les processeurs, les blocs de mémoire centrale. Dans certaines machines cette aptitude à la reconfiguration dynamique a été poussée jusqu'à la possibilité de couper le système en plusieurs systèmes indépendants: rôle de l'unité de partitionnement sur les 1100 de Sperry.

16.7 – Arrêt du système.

Un système peut s'arrêter de deux façons:
- normalement (proprement, orderly shutdown) à la suite d'une commande de l'opérateur,
- anormalement ("plantage", crash), de façon non planifiée, non voulue. Généralement on effectue alors un vidage (dump) de l'état de la mémoire afin de rechercher la cause du mauvais fonctionnement.

chapitre 17

gestion des performances

Un système informatique est destiné à offrir des services à ses utilisateurs et la qualité de ces services sera d'autant plus appréciée qu'ils seront performants. Par performances nous entendons ici:
- la **vitesse** à laquelle ces services sont fournis à l'utilisateur, par exemple le temps séparant la question qu'il formule de la réponse reçue,
- le taux d'**utilisation des ressources** qui fera d'autant plus baisser le coût du service fourni qu'il est élevé.

17.1 – La démarche "performances".

Un système informatique a des **objectifs** de performance qui sont définis au moment de sa conception puis affinés lors de sa réalisation. Lorsque le système est mis en exploitation, en "production", il convient de vérifier si ces objectifs prédéfinis sont atteints. Il faut donc procéder à des **mesures**.

Ces mesures nécessitent l'activation de fonctions et/ou outils fournis avec le système. Elles peuvent conduire à se doter d'**outils** particuliers au problème des performances. Il y aura alors une étude qui conduira à poser le problème, tester des outils, établir un cahier des charges, puis choisir l'outil jugé le plus adapté au problème à résoudre, pour finalement l'installer et le mettre en service.

Une fois les mesures collectées on dispose de résultats sous différentes formes: tableaux, graphes, camemberts, etc., qu'il faudra **interpréter** ou dont il faudra compléter l'interprétation proposée.

Ces interprétations basées sur les mesures et la connaissance du système mesuré conduisent généralement à des **optimisations** qu'il faudra ensuite valider par une nouvelle campagne de mesures. Nous distinguerons deux types de mesures:
- les mesures **continues**, réalisées systématiquement, par exemple la collecte des statistiques destinées à la facturation du système informatique;
- les mesures **exceptionnelles** mises en oeuvre dans un but précis. Elles constituent ce que nous avons appelé une **campagne de mesure**. En temps normal les informations collectées dans un but de métrolo-

gie sont maintenues à un strict minimum afin de ne pas surcharger les ressources. Mesurer met en effet en oeuvre le processeur, la mémoire, les supports de stockage, les canaux, créant ainsi une charge non productive.

17.2 - Les mesures en continu.

Un système informatique gère au minimum deux fichiers d'événements:

- l'un contenant des informations permettant de reconstituer a posteriori la charge soumise, d'établir les factures destinées aux utilisateurs. C'est le rôle du fichier **comptable** (account) appelé SCF sous GCOS3, SMF sous MVS, ACCOUNTING.DAT sous VMS;
- l'autre mémorisant les notifications de fonctionnement anormal comme les pannes, les taux d'erreur, etc.

Les informations comptables permettent de mesurer le taux d'occupation de certaines ressources comme le(s) processeur(s), la mémoire, les disques, etc., mais ce n'est pas leur but initial.

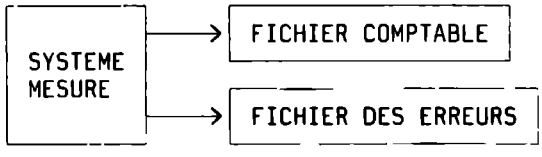


Figure 17a: les deux fichiers que l'on trouve généralement sur les ordinateurs.

L'utilisation du **processeur** peut être ainsi reconstituée avec les limitations suivantes:

- dans certains systèmes, seul le temps dit "**utilisateur**" est donné, à l'exclusion du temps "système" qui est le temps consommé par les instructions du système d'exploitation. Dans d'autres SE seul le temps en mode "**esclave**" (mode problème chez IBM) figurera dans le fichier comptable, à l'exclusion du temps passé en mode "maître" (ou superviseur). Ces temps "système" ou "maître", différents l'un de l'autre d'ailleurs car toutes les instructions du SE ne se déroulent pas en mode maître", sont parfois appelés "**déperdition**" (overhead) pour traduire le sentiment que cette consommation n'est pas productive, mais consommée par le SE pour gérer les requêtes et les ressources. Ils représentent cependant certaines actions qui devraient être réalisées par les programmes des utilisateurs si le SE ne s'en chargeait: cas des E/S physiques par exemple. Une partie de ces temps correspond réellement à de la déperdition: [VMS84E] souligne que la déperdition décroît lorsque la tranche de temps du distributeur croît; dans un multiprocesseur, un processeur qui boucle (attente active) dans l'attente qu'un autre processeur libère une ressource à un seul accès ("gating" de GCOS3), représente aussi une déperdition;
- le temps processeur (appelé aussi "temps CPU") n'est pas toujours **correct** (il ne reflète pas la réalité physique), il n'est pas toujours **fidèle**, plusieurs passages de la même étape (de la même commande) pouvant conduire à des temps processeur différents, en

particulier dans les multiprocesseurs.
Comme toutes les reconstructions a posteriori l'utilisation du processeur déduite du contenu du fichier comptable est sujette aux problèmes de lissage que nous illustrerons par l'exemple de la figure 17b où deux programmes sont exécutés durant une période de 20 minutes: le programme P1 se déroule durant les 10 premières minutes et consomme 8 minutes de processeur, le programme P2 démarre en même temps, dure 20 minutes et consomme 10 minutes de processeur. Du fichier comptable on déduira que P1 charge le processeur à 80 % (8 minutes sur 10) pendant les 10 premières minutes, et que P2 le charge à 50 % (10 minutes sur 20) pendant 20 minutes. On constate que le processeur est chargé à 130 % pendant les 10 premières minutes, ce qui est bien évidemment impossible. Cet exemple simpliste avait pour seul but de souligner le caractère limité de ces informations comptables, valables surtout sur de longues périodes lorsqu'on les utilise pour mesurer le taux d'occupation des ressources.

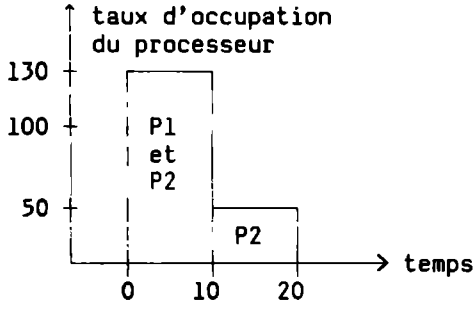


Figure 17b: reconstitution de l'utilisation du processeur à partir du fichier comptable.

Le **fichier des erreurs**, a priori étranger aux performances, est cependant susceptible de fournir des explications à certaines baisses des performances: si le taux d'erreur sur une bande magnétique est élevé, il y aura de nombreuses tentatives supplémentaires d'entrée-sortie et le débit global s'en ressentira. Il importe donc de vérifier que les ressources fonctionnent correctement. Le fichier comptable donne un certain nombre d'informations concernant les erreurs.

Après le processeur, la **mémoire** est la ressource la plus importante. Les fichiers comptables donnent la taille par la mémoire pour chaque étape. Quand cette taille varie en cours d'exécution cette information devient d'une utilisation limitée. Sous GCOS3 on a ainsi le produit de la taille mémoire multipliée par le temps de résidence en mémoire, ce qui permet d'en calculer la taille moyenne une fois le temps de vidage (swapping) retranché au temps d'exécution (elapsed time).

L'activité d'E/S se trouve reflétée dans le fichier comptable sous forme de:
- temps d'E/S, global et par fichier (GCOS3),
- nombre d'E/S physiques (nombre d'EXCP chez IBM),
- nombre d'unités de données transférées: cartes, lignes, blocs,
- liste des périphériques/fichiers accédés.

Ces mesures en continu avec effet a posteriori (les données collectées sont ensuite traitées, réduites, plusieurs heures ou jours après le début de la période mesurée) permettent un suivi **global** de l'activité des ressources et sont principalement destinées à répercuter les coûts d'utilisation du système informatique en utilisant des formules où chaque type de ressource est pondéré par un facteur de coût du type:

$$\text{coût facturé} = A * t_p + B * m + C * t_{ed} + D * t_{ei} + \ldots$$

où A est le coût unitaire du temps processeur, B celui de la mémoire, C celui des E/S disque, D celui des E/S imprimante, etc. C'est ainsi que l'on parlera d'unité URSH dans le monde de GCOS3, de CRU (Computer Resource Unit) chez BCS (Boeing Computer Services, cf. [HOL78]):

$$CRU = a * t_p + b * RUT * m + c * t_{ed} + e * n_{es} + f * RUT * n_{dt} + g * RUT$$

avec t_p le temps processeur, m la taille mémoire, t_{ed} le temps d'entrée/sortie sur les disques, n_{es} le nombre d'E/S, n_{db} le nombre de dérouleurs de bande, RUT (Resource Unit Time).

Les constructeurs fournissent le format des fichiers comptables et offrent parfois un programme de réduction (GSEP de GCOS3). Plus généralement l'utilisateur écrit son propre programme de traitement des informations comptables, en prend un qui figure dans la bibliothèque du club des utilisateurs, ou même l'achète auprès d'une SSII (cas de Johnson pour DOS d'IBM).

17.3 – Les mesures indispensables.

Selon les objectifs que se fixent les responsables du système informatique, des mesures complémentaires peuvent se révéler nécessaires sur une base continue, les précédentes étant considérées comme un minimum indispensable. Les outils les plus classiques correspondent à:

- l'ajout de nouveaux types d'enregistrement dans le fichier comptable. Bull a ainsi commercialisé dans la seconde moitié des années 70 une option permettant d'obtenir le temps processeur "système" dans le fichier comptable; RMF sous MVS ajoute ses propres articles (type 7X) dans le fichier comptable SMF;
- une approche **temps réel** et non plus **différée** comme celle du fichier comptable. Les informations collectées sont alors aussitôt affichées sur un écran.

Les premiers outils temps réel ajoutés sont généralement destinés aux opérateurs/pupitreurs (cf. VIDEO sous GCOS3) et/ou responsables du système, mais rarement dans un contexte de mesure en continu pour ces derniers. Ces outils (**logimètres**) sont basés sur deux techniques de collecte:

- **événementielle**: toute occurrence d'une information qui fait l'objet de la mesure est prise en compte;
- **périodique**: selon une fréquence prédéterminée, les informations requises sont prélevées, traitées et affichées en temps réel. Plus le nombre d'informations suivies est élevé, plus la fréquence d'échantillonnage est grande, plus le coût de la mesure sera important.

Dans le cas d'un suivi global temps réel par les opérateurs, on se limitera à:
- l'état des appareils,
- le taux d'occupation des ressources,
- la liste des travaux en cours, l'état des différentes étapes, les utilisateurs connectés,
- la description globale de la charge déjà traitée,
- la description globale de la charge en attente,
- l'état du spoule de sortie (SYSOUT).

17.4 – Mesures exceptionnelles.

Lorsque l'on procède à une campagne de mesures avec un objectif plus précis que la simple surveillance globale, on active des outils de mesure supplémentaires. Selon l'objectif on utilisera tel ou tel outil, avec tel ou tel paramétrage.

Supposons que le problème à résoudre soit l'optimisation (tuning) globale du système. Il convient alors de rechercher:
- s'il y a des ressources physiques bloquantes (des "goulots d'étranglement", bottleneck),
- si le système d'exploitation est bien adapté (généré, paramétré) à la charge à traiter,
- si certaines applications peuvent être améliorées.

La première chose à faire est d'analyser le plus gros consommateur de ressources, à savoir les applications, d'en étudier le comportement dynamique, c'est-à-dire en cours d'exécution.

[VMS84E] note d'ailleurs que "Digital pense qu'il y a rarement besoin d'optimiser un système VAX/VMS ... Plusieurs raisons militent en ce sens ... D'une façon limitée le système s'adapte automatiquement durant le fonctionnement: optimisation des tailles du pool dynamique non paginé, de l'«espace vital» (working set), des listes de pages libres et modifiées". Pour mesurer ces applications on dispose d'outils ("logimètres"):
- les uns, externes à l'application et basés sur une technique d'échantillonnage (sampling), interrompent à fréquence fixée le programme étudié (cas de SPM, Slave Program Monitor sous GCOS3) et relèvent à chaque interruption l'adresse (donnée par le compteur ordinal CO) de l'instruction ainsi interrompue. Après terminaison du programme, les adresses relevées sont analysées et présentées sous forme d'histogramme qui met immédiatement en évidence les séquences les plus souvent exécutées du programme. La table de chargement (load map) fournie par l'éditeur de liens (linker, cf. tome V) permet de connaître le temps passé dans chaque sous-programme. Le programme ainsi analysé n'est en rien modifié au niveau de sa forme source (PS) ou de sa fome exécutable (PE). Si les résultats doivent être corrélés avec le niveau source (histogramme faisant apparaître les numéros de ligne du programme source et non plus les adresses physiques des instructions) il faut activer une option au niveau du compilateur. Plus la fréquence d'échantillonnage est élevée, plus les résultats sont précis, mais le coût croît aussi en conséquence;

- d'autres, internes à l'application mesurée, supposent une modification du source et/ou l'ajout d'instructions au niveau de la compilation, modifiant de toute façon la forme exécutable. Lors du déroulement du programme, ces instructions supplémentaires incrémentent des compteurs à chacune de leur exécution. Après déroulement du programme ces compteurs sont vidés et l'on dispose du compte exact du nombre d'exécutions de chaque séquence. Ce genre d'outil, indépendant de tout ordinateur, est facile à réaliser.

Le comptage donne cependant une information moins significative que l'échantillonnage qui tend à refléter exactement les rapports de consommation du processeur lorsque la fréquence est suffisamment élevée. De plus l'échantillonnage ne se limite pas au code utilisateur, mais à tous les sous-programmes ajoutés au moment de l'édition de liens. Pour un programme faisant beaucoup d'entrées/sorties on verra de suite que la majeure partie du temps consommé l'est par les sous-programmes des méthodes d'accès.

Nous ne nous étendrons pas plus avant sur la méthode d'analyse des applications qui doit s'opérer sur les plus grosses applications (règle des 80/20), désignées comme telles par l'analyse du fichier comptable.

Une fois les applications considérées comme optimisées (ou non optimisables) du point de vue des performances, il convient:

- d'étudier la charge des différentes ressources du système afin de déterminer celle qui est bloquante et ralentit l'ensemble,
- d'optimiser le système d'exploitation, ou
- de l'adapter aux ressources disponibles.

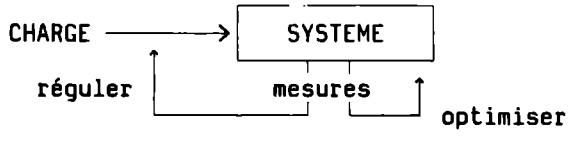


Figure 17d: le double rôle des mesures.

Une ressource peut être saturée sans qu'il faille en conclure automatiquement qu'elle doit être augmentée car cette saturation peut provenir d'un autre phénomène. On trouvera dans [MON80] une étude concernant un 43X1 sous DOS/VSE où la saturation du canal disque a été résolue en utilisant un logiciel conservant en mémoire centrale les modules système les plus souvent utilisés.

Dans d'autres cas on pourra trouver un ralentissement provoqué par les accès qui se concentrent sur un seul disque, et il suffit parfois de réorganiser les fichiers sur le disque (optimiser les déplacements des bras) ou de les redistribuer sur plusieurs disques pour surmonter le problème.

Les constructeurs des SE fournissent généralement un manuel indiquant comment aborder la recherche des goulots d'étranglement et quels outils utiliser (cf. [VMS84E]).

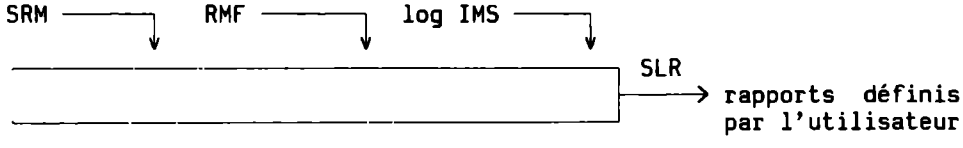
Les constructeurs (utilitaire ACCOUNTING sur VMS), les SSII fournissent une panoplie d'outils basés sur des compteurs internes au système d'exploitation, sur des mécanismes de trace initialement prévus

pour des besoins de diagnostic en cas de problème, etc. Ces logimètres logiciels ont le double défaut de consommer des ressources et de perturber le système mesuré; ils permettent par contre de mieux cerner certains événements qui ne sont pas visibles de l'extérieur. Leurs concurrents, ou leurs compléments selon les points de vue, les logimètres **matériels** (hardware monitor), ne perturbent pas le système mesuré mais ne voient que les signaux émis au niveau du matériel; ils donnent cependant des mesures absolues ne souffrant aucune contestation.

OUTILS
- INTERNES ou EXTERNES par rapport au système mesuré
- collecte par COMPTAGE ou ECHANTILLONNAGE
- affichage en TEMPS REEL ou en TEMPS DIFFERE
- au niveau APPLICATIF ou SYSTEME

Figure 17e: caractéristiques des logimètres.

Notons que les outils de mesure sont de plus en plus intégrés au SE et capables d'offrir certaines des fonctions suivantes:
- bâtir une base de données (SLR d'IBM, MICS de Morino Associates),



- confronter ces mesures avec les objectifs de performance préalablement définis et émettre des messages lorsque des conditions critiques sont atteintes,
- réagir automatiquement sur la prise en compte de la charge de travail en attente (aspect de contre-réaction), en diminuant par exemple le taux de multiprogrammation si la charge processeur est excessive, si la pagination est trop élevée (cas des "load balancers" de SRM dans MVS).

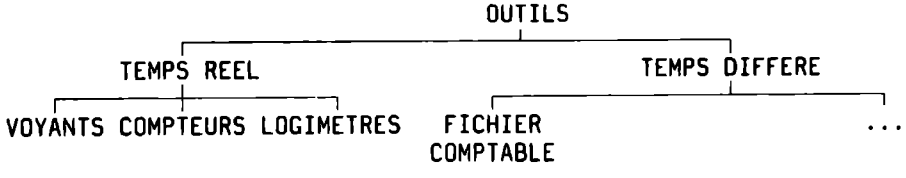


Figure 17f: classification des outils de mesure.

17.5 – Suivi des performances par les constructeurs.

Lorsqu'ils annoncent de nouvelles versions de leurs logiciels, les constructeurs publient des mesures comparatives par rapport aux anciennes versions pour indiquer les effets de ces améliorations. Une amélioration fonctionnelle peut provoquer des baisses de performance.

Ainsi [BAR82] compare deux versions successives de MVS/SP, dans trois modes de fonctionnement bien typés: en traitement par lots, en temps partagé (TSO), en transactionnel (IMS).

Comme nous l'avons mentionné dans le tome I, les constructeurs ont de grandes difficultés à trouver des charges représentatives de ce que font réellement les utilisateurs. Pour s'en approcher, ils utilisent des charges synthétiques qu'ils repassent à chaque évolution de logiciel dans des conditions identiques ou aussi comparables que possible.

Dans le cas du traitement par lots, [BAR82] utilise deux charges de travail (workload):

- l'une appelée PCSV avec 10 travaux de deux à sept étapes comprenant principalement du Cobol et destinées à représenter un environnement de gestion typique sous MVS:

NBR ETAPES	%	DESCRIPTION (% d'un 4341-2)
7	31,2	Cobol CLG et tri (crée fichier 5000 articles)
5	4,5	Cobol CLG
5	5,9	↑ Cobol, exécution, mise à jour fichier,
5	7,2	nombre d'articles différents selon les
5	8,3	travaux: de 1000 à 10000
5	9,7	
5	17,2	↓
2	5,1	assemblages de petites CSECT
3	5,4	↑ Cobol CLG, création de 3 fichiers de
3	5,4	↓ 1000 articles

- l'autre appelée CB80 (Commercial Batch 1980) comprend 30 travaux plus proches des environnements de gestion réellement observés par IBM. Elle est reprise dans [BAU83] mais avec une définition quelque peu différente: 102 étapes au lieu de 52, 191 fichiers au lieu de 130:

NOMBRE ETAPES	NOMBRE FICHIERS	METHODE D'ACCES	TAILLE BLOC	DESCRIPTION
30				programmes synthétiques
12				Cobol, exécution
4	75	QSAM	2504	tri Iceman
3	22	BSAM	1816	édition de liens
2	17	BDAM	2750	Cobol, compilation
1	16	VSAM	3068	assembleur

Pour la charge en temps partagé [BAR82] utilise treize "scripts" différents repris aussi dans [BAU83]:

1. édition de texte, allocation d'un fichier, suppression du fichier,
2. édition d'un source en langage d'assemblage (deux transactions plein écran) et suppression (cancel) de l'édition,
3. assemblage via SPF et chargement/exécution par une CLIST,
4. balayage (browse) d'un listing de PLIFOR,
5. édition en insérant ligne à ligne dans un programme Cobol puis suppression de l'édition,
6. utilisation du "prompter" Cobol pour compiler, éditer les liens, puis utilisation de TESTCOB; allocation et suppression de trois fichiers partitionnés,
7. copie et impression d'un fichier par des menus SPF,
8. tri SPF et déconnexion (logoff), puis connexion (login) et commande SPF,
9. édition d'un source PL/I (deux transactions plein écran) puis suppression de l'édition,
10. PL/I interactif et ré-utilisation ou allocation d'un fichier liste,
11. édition d'un travail, soumission (submit), suppression de l'édition, travail mis dans l'état bloqué (hold),
12. exécution avant-plan (foreground) de Script/VS (formatteur de texte), puis visualisation (scroll) de la sortie et impression du fichier,
13. Entrée de six demi-écrans de Script/VS puis suppression de l'édition.

TRAN-SACTION	APPELS VSAM	READ	READ NEXT	ARTICLE: AJOUT	MODIF	SUPPRES	%	DESCRIPTION
1	3	2			1		8	.banque
2	10	8		2			8	banque
3	2	1			1		3	.réservation
4	2	1		1			3	hôtel
5	0						3	.inventaire
6	17	1	16				6	et
7	14	5	4	3	2		8	suivi
8	3	1	2				5	.entrée
9	3	1		1	1		5	de
10	9	9					5	commandes
11	22	9		4	9		5	
12	18	8	10				10	.spécification
13	34	2	32				9	de produits
14	18	1	8		9		5	.contrôle
15	9			9			3	de
16	3	1	1			1	10	stock
17	0						4	.caisse

Tableau 17g: charge transactionnelle de référence chez IBM.

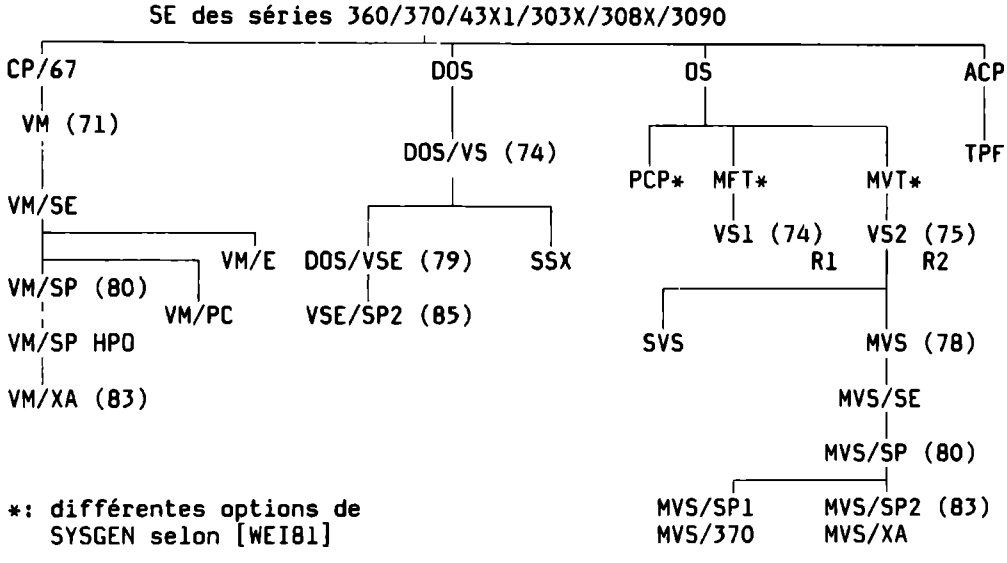
Les terminaux simulés déroulent les treize scripts ci-dessus, le comportement des utilisateurs est représenté par leur temps de réflexion (think time). Dans les mesures de [BAR82] ce temps de réflexion est celui observé sur une machine de développement utilisée à l'intérieur d'un laboratoire d'IBM. Les terminaux interactifs sont simulés par un programme appelé LDTS (Local Display Terminal Simulator) qui tourne dans le système mesuré et le perturbe donc quelque peu. IBM commercialise un générateur de charge externe (TPNS) dont la mise en oeuvre est plus lourde et requiert plus de ressources matérielles. Bull dispose aussi d'outils analogues: TDSLG (TDS Load Generator) joue un rôle analogue à LDTS dans l'environnement transactionnel, Cuesta et OST sont des équivalents de TPNS. Dans [CLA85] DEC utilise un PDP-11 doté d'un logiciel RTE (Remote Terminal Emulator) pour simuler une charge.

Pour le **transactionnel** [BAR82] utilise une charge appelée DSW (Data Systems Workload) développée en 1979. [BAU83] l'utilise aussi pour IMS et CICS. Elle comprend trente-quatre transactions dont dix-sept ont une pondération définie pour coller à une enquête réalisée par les groupes d'utilisateurs Guide/Share. L'utilisation sous CICS, avec des fichiers VSAM est décrite dans [BAU83] (tableau 17g).

BIBLIOTHÈQUE
Université du Québec à Rimouski

annexe A

SE d'IBM sur 370



*: différentes options de SYSGEN selon [WEI81]

1964: les 360
1970: les 360
1977: les 303X
1979: les 43X1
1980: les 308X
1985: les 3090

Après les annonces IBM du 11 février 1986, les DOS et OS étaient supportés comme suit par les différents modèles d'ordinateurs:

	4331-11-4321	4361-3	4361-4	4361-5	4381-11	4381-12	4381-13	4381-14	3083-EX	3083-BX	3083-JX	3083-KX	3083-QX	3083-CX	3083-EX	3083-BX	3083-JX	3083-GX	3083-KX	3083-QX	3090-1100	3090-1500	3090-2000	3090-4000
MVS/370				*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
VM/XA					*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
MVS/XA					*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
DOS/VSE	*	*	*	*	*	*	*	*																
SSX/VSE	*	*	*	*	*																			
VM/370	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

Le Monde Informatique du 30 avril 1984 parlait de 1200 utilisateurs VSE en France, 6000 en Europe et 12000 dans le monde. Selon [LMI86], 11 % des sites MVS ont XA en fin 84. Ce taux devrait passer à 62 % fin 86, représentant 20 000 sites. Selon [ORD86], la progression de MVS/XA entre 85 et 86 aux Etats-Unis est reflétée ci-dessous par le taux de pénétration sur différents ordinateurs:

EN % EN	MAI 85	MAI 86
4381	4,8	10
3081	26	50
3084	6,2	80
Amdahl 580	3,3	32
NAS	0	28

annexe B

lexique

area: zone
background: arrière-plan
balance: équilibrer
buffer: tampon
cache: antémémoire
catalog: répertoire
data set: fichier
device: appareil
directory: répertoire
dispatcher: distributeur
extent: extension
file: fichier
foreground: avant-plan
frame: cadre
hidden: caché.
idle: inactif.
job: travail
kernel: noyau
label: étiquette
load: charge
lock: verrou
log: connexion, journal
map: plaquer, carte
overhead: déperdition
pipe: tube
primary: principal, primaire
process: processus, traiter
qualifier: qualifieur
queue: file d'attente
random: aléatoire
release: version
roll: vider
round robin: tourniquet
scheduler: planificateur
slot: case
spool: spoule
stack: pile
step: étape
swap: vider
switch: commuter
task: tâche
thrashing: emballement
token: jeton.
translation: traduction
trimming: réduction
tuning: optimiser (régler).
update: mise à jour
workload: charge de travail

annexe C

abréviations

ABEND: ABnormal END.
ACF: Advanced Communications Function (IBM).
ACK: Advanced Communication Kernel (Philips).
ACP: Airline Control Program (IBM, devenu TPF).
ADI: Agence De l'Informatique.
AF: Advanced Functions (VSE/AF).
AL: Adresse Logique.
AM: Access Method.
AMS: AM Services *ou* Availability Maintenability Serviceability.
AOS: Advanced OS.
AP: Adresse Physique.
APF: Authorized Program Facility (MVS).
API: Application Program Interface (IBM).
ASM: Auxiliary Storage Manager (MVS).
ASP: ASymmetric multiProcessing.
AT&T: American telegraph and Telephone.
AV: Adresse Virtuelle.
AWSA: Automatic WS Adjustment (VMS).
BàL: Boîte à Lettre.
BDOS: Basic DOS.
BDT: Bulk Data Transfer (IBM).
BIOS: Basic I/O System (MS-DOS, iRMX86).
BSC: Binary Synchronous Communication (IBM).
BSD: Berkeley Software Distribution (Unix).
CA: Control Area (UFAS, VSAM).
CALC: Core ALoCator (GCOS3).
CAW: Channel Address Word (IBM).
CCITT: Comité Consultatif International Télégraphique et Téléphonique.
CCP/M: Concurrent CP/M (devenu Concurrent DOS).
CDC: Control Data Corporation.
CDS: Code and Data Separated (RTE-A).
CI: Control Interval.
CICS: Customer Information Control System (IBM).
CLG: Compile, Link and Go.
CMS: Conversational Monitor System (VM).
CNET: Centre National D'Etudes des Télécommunications.
CNM: Communications Network Management (IBM).
CO: Compteur Ordinal.
CODASYL: COmmittee on DAta base SYstem Languages.
CP: Control Program.
CPF: Control Program Facility (S/38).
CPL: Current Privilege Level (iAPX).
CP/M: Control Program for Microcomputers (DRI).
CSA: Common Service Area (MVS).
CSECT: Code SECTion (IBM).
CTC: Channel To Channel (IBM).
CTOS: Convergent Technology OS (STARSYS chez Bull).
DAM: Direct AM.
DAS: Dual Address Space (IBM/370).
DASD: Direct Access Storage Device.
DAT: Dynamic Address Translation.
DB/DC: Data Base/Data Communications.
DBMS: Data Base Management System.
DBTG: Data Base Task Group (CODASYL).
DB2: Data Base 2 (IBM).
DCB: Device Control Block.
DCL: DEC Control Language.
DCS: Dynamic Channel Subsystem (IBM XA).
DDR: Dynamic Device Reconfiguration (JES3).
DEB: Data Extent Block (IBM).
DEC: Digital Equipment Corporation.
DG: Data General.
DITTO: Data Interfile Transfer, Testing and Operations (VSE).
DJC: Dependent Job Scheduling.
DLAT: Directory Look-Aside Table (IBM).
DL/I: Data Language number 1.
DLIB: Distribution LIBrary (IBM).
DMERT: Duplex Multiple Environment Real Time (AT&T).
DOS: Disk Operating System.
DPCX: Distributed Processing Control eXecutive (8100).
DPL: Descriptor Privilege Level (iAPX).
DPPX: Distributed Processing Programming eXecutive (8100).
DPS: Distributed Processing System.
DRI: Digital Research Inc.
DSA: Distributed System Architecture (Bull).

ECPS: Extended Control Program Support (IBM).
EdL: Editeur de Liens.
EdT: Editeur de Texte.
EIOS: Extended IOS (iRMX).
EMA: Extended Memory Area (HP1000).
EP: Emulation Program (IBM/37X5).
ES: Extended Storage (BTAM-ES).
E/S: Entrée/Sortie.
EVMA: Expanded VMA (IBM).
FAT: File Allocation Table.
FCB: File Control Block.
FCFS: First Come First Served.
FEP: Front End Processor.
FIFO: First In First Out.
FMS: File Management System.
GCOS: General Comprehensive OS (Bull).
GESP: Gestionnaire des Entrées/Sorties Physiques.
GFRC: General File (GCOS3).
GIN: General INput (GCOS3).
GMC: Gestionnaire de la Mémoire Centrale.
GMV: Gestionnaire de la Mémoire Virtuelle.
Go: Giga-octet (2^{30}).
GOT: General OUtput (GCOS3).
GPL: Gestionnaire des Périphériques Lents.
GRTS: (GCOS3).
GTF: Generalized Trace Facility (MVS).
GTM: Generalized Transaction Monitor (IBM).
HASP: Houston Automatic Spool Program (IBM).
HCM: Hard Core Monitor (GCOS3).
HP: Hewlett-Packard.
HPO: High Performance Option (IBM).
IBM: International Business Machine.
ICCF: Interactive Computing and control Facility (VSE).
ICL: International Computer Limited.
IDC: International Data Corporation.
IDS: Integrated Data Store.
IH: Interrupt Handler.
IML: Initial Microcode Load (IBM).
IMS: Information Management System (IBM). (Bull).
I/O: Input/Output.
IOS: Input Output Supervisor.
IPL: Initial Program Loading (IBM).
IPO: Installation Productivity Option.
IPS: Installation performance Specification (MVS).
IS: Integrated System (VM/IS).
ISAM: Indexed Sequential AM.
ISC: Inter System Coupling (IBM/CICS).
ISO: International Standards Organization (OSI en français).
ISP: Indexed Sequential Processor (GCOS3).
ISPF: Interactive System Productivity Facility (IBM).
ISV: Internal Service Value.
IX: Interactive eXecutive (Unix sur 370).
JCL: Job Control Language.
JES: Job Entry System.
JSAM: JES3 Spool AM.
Km: Kilo-mot (1024).
Ko: Kilo-octet (1024).
LIOCS: Logical IOCS.
LPA: Link Pack Area (OS d'IBM).
LPDS: Local PDS (MVS).
LRU: Least Recently Used.
LU: Logical Unit (RTE-A et SNA).
MAE: Machine A Ecrire.
MC: Mémoire Centrale.
MCP: Master Control Program (Burroughs).
MDF: (Amdahl).
MEM: MEmoire Morte.
MEV: MEmoire Vive.
MF/I: Measurement Facility I (MVS).
MFT: Multiprogramming with a Fixed number of Partitions (IBM).
MLPA: Modified LPA (MVS).
MME: Master Mode Entry (DPS8).
MMU: Memory Management Unit.
Mo: Méga-octet (2^{20}).
MPL: MultiProgramming Load.
MP/M: Multiprogramming operating system for Microcomputers (DRI).
MPE: (HP3000).
MPP: Message processing Program (IMS).
MPR: Message Processing Region (IMS).
ms: milliseconde.
MSC: Multiple System Coupling (IBM/IMS).
MS-DOS: MicroSoft Disk DOS.
MS-NET: MicroSoft NETwork.
MSS: Mass Storage System.
MSX: MicoSoft eXtended.
MULTICS: MULtiplexed Information and Computing Service.
MVS: Multiple Virtual Storage.
MVT: Multiprogramming with a Variable number of Tasks.
NAU: Network Addressable Unit (IBM).
NCCF: Network Communications Control Facility (IBM, SNA).
NCP: Network Control Program (IBM).
NETBIOS: NETwork BIOS (PC).
NIP: Nucleus Initialization Program (MVS).
NJE: Network Job Entry (JES2).
NJI: Network Job Interface (VM).
NJP: Network Job Processing (JES3).
NLDM: Network Logical Data Manager (NCCF).
NOS/VE: Network OS *ou* /Virtual Environment (CDC).
NPA: network Performance Aid (NCCF).
NPDA: Network Problem Determination Application (NCCF).
NPM: Network Performance Management (IBM).
ns: nano-seconde.
OCCF: Operator Communication Control Facility (IBM).
OCL: Operator Control Language.
OLTEP: OnLine Test Executive Program .SE).
OS: Operating System.
OSI: Open System Interconnection.
OSNS: Open Systems Network Support (IBM).
OTSS: Open systems Transport and Session Support (IBM).
PALC: Peripheral ALloCator (GCOS3).
PC: Processeur Central *ou* Personal Computer.
PCB: Process Control Block.
PC-DOS: IBM Personal Computer DOS.
PEP: Partitioned Emulation Program (IBM/37X5).
PFT: Page Frame Table (MVS).

PDS: Page Data Set (MVS) *ou* Partitioned Data Set.
PE: Programme Exécutable.
PID: Prise ISO/DSA (Bull).
PKM: PSW Key Mask (370/DAS).
PIOCS: Physical IOCS.
PLPA: Pageable LPA (MVS).
PMA: Preferred Machine Assist (VM).
POWER: Priority Output Writer Execution processors, and input Readers (DOS/VS).
PPS: Produit Programme du Système (S/36 IBM).
PSW: Processor Status Word.
PTE: Page Table Entry.
PTF: Program Temporary Fix (IBM).
PU: Physical Unit (SNA).
QIO: Queued I/O (VMS).
QSAM: Queued SAM (IBM).
RB: Remote Batch.
RCP: Resource Control Package.
RCT: Region Control Task.
RGIN: Remote GIN (GCOS3).
RI: Registre Instruction.
RIM: Resource Initialisation Module (MVS).
RJE: Remote Job Entry (JES2).
RJP: Remote Job Processing (JES3).
RMF: Resource Measurement Facility (MVS).
RMS: Record Management System (VMS).
ROCF: Remote Operator Control Facility (IBM).
RPL: Requested Privilege Level (iAPX) *ou* Remote Program Loading (IBM).
RPQ: Request for Price Quotation.
RPS: Real time Programming System (S/1).
RSM: Real Storage Management (MVS).
RSCS: Remote Spooling Communications Subsystem (VM).
RT: Real Time *ou* Registre de Traduction.
RTAM: Real Terminal AM (IBM).
RTE-A: (HP1000).
RTOS: Real Time OS.
RTR: Real Time Reliable.
SAM: Sequential AM *ou* System Available Memory (RTE-A).
SCF: System Collection File (GCOS3)
SDLC: Synchronous Data link Control (IBM).
SDS: Swap Data Set (MVS).
SE: Système d'Exploitation *ou* System Extension (MVS/SE).
SED: SE Distribué.
SER: SE Réseau.
SGF: Système de Gestion de Fichier.
SSCI: Société de Service et de Conseil en Informatique, devenu SSII.
SSCP: System Services Control Point (VTAM).
SSI: SubSystem Interface.
SSII: Société de Service et d'Ingéniérie en Informatoque.
SSX: Small System eXecutive.
SVA: Shared Virtual Area (DOS/VS).
SVC: SuperVisor Call (IBM).
SVID: System V Interface definition (Unix).
SVS: Single Virtual Space (IBM).
SYSIN: SYStem INput.
SYSOUT: SYStem OUTput.
TCAM: TeleCommunication AM.
TCB: Task Control Block.
TDS: Transaction Driven System.
TdP: Table des Pages.
TdS: Table des Segments.
TdT: Traitement de Texte.
TLB: Translation Look aside Buffer.
To: Tétra-octet (2^{40}).
TPA: Transient Program Area (CP/M).
TPF: Transactions Processing facility (IBM).
TPNS: TeleProcessing Network Simulator (IBM).
TR: Temps Réel.
TSO: Time Sharing Option.
TSS: Time Sharing System (GCOS3) *ou* Task State Segment (iAPX).
TTY: TeleTYpe.
UFAS: Unified File Access System.
URSH: (Bull).
USAM: User Spool AM (JES3).
VAX: Virtaul Address eXtension (DEC).
VIO: Virtual I/O.
VM: Virtual Machine (IBM).
VMA: Virtual Memory Area (RTE-A) *ou* Virtual Machine Assist (VM).
VMS: Virtual Memory System (VAX).
VS: Virtual Storage.
VSAM: VirtuaL Storage AM (IBM).
VSE: VS Extended (IBM).
VSM: Virtual Storage Management (MVS).
VSPC: VS Personal Computing (IBM).
VTAM: Virtual Telecommunication Access Method (IBM).
VTOC: Volume Table Of Contents.
WS: Working Set.
WSP: Working SPace (GCOS8).
XPT:eXternal Page Table (MVS).

annexe D

références

ACK8I Le système d'exploitation Prologue et les télécommunications.
JM. Ackermann.
Minis et Micros numéro 210 p. 69-71.

AME83 Security kernel design and implementation: an introduction.
S.R. Ames, M. Gasser, R.R. Shell.
Computer vol 16 numéro 7, juillet 83 p. 14-22.

AND75 Tuning a virtual storage system.
H.A. Anderson, M. Reiser, G.L. Galati.
IBM Systems Journal numéro 3, 1975, p. 246.

AND81 Operating Systems.
D.A. Anderson (Bell Labs).
Computer vol 14 juin 81 p. 69-82.

AND86 System/38 grows up.
D. Andrews, F. Manteghian.
Computerworld In Depth, 21avr. 1986 p. 79-82.

ATT79 Virtual Control Storage - Security measures in VM/370
C.R. Attanasio
IBM Systems Journal numéro 1, 1979, p. 93

AUS73 Functional structure of IBM Virtual Storage Operating Systems: influences of dynamic address
M.A. Auslander, J.F. Joffe
IBM System Journal numéro 4, 1973, p. 368.

BAI84 Hardware, software trends expand multiuser system performance.
Ch. Bailey.
Mini-Micro Systems juin 84 p. 71-75.

BAL81 Le programme E10.S-TSS.5.
M. Ballard, M. Martin (CIT-Alcatel).
Commutation & Tansmission numéro 3, 1981 p. 99-112.

BAN80 The impact of distribution on OS design.
J.S. Banino, A. Caristan, M. Guillemont, G. Morisset, H. Zimmermann.
CHORUS project, INRIA.

BAN80A CHORUS- an architecture for distributed systems.
J.S. Banino, A. Caristan, M. Guillemont, G. Morisset, H. Zimmermann.
INRIA rapport de recherche numéro 52, nov. 1980.

BAN85 Ease of use in application development.
K. Bandat (IBM).
SEAS proceedings 23-27 sept. 85, p.257-275.

BAR77 The Tandem 16: a "non stop" OS.
J.F. Bartlett.
Tandem.

BAR82 Comparison of the MVS/SP version 1 release 1 to MVS/SP version 1 release 3 on the 4341 model group 2.
J. Barnes, P.R. Dorn, L. Fadel, R.J. de Jonge, E.A. Wetzelberg.
IBM Technical Bulletin GG22-9272-00, janv. 82.

BAR84 Unix - How important is it?
D.F. Barlow, N.S. Zimbel.
Datamation août 84 p. 90-101.

BAS82 Pipelining and new OS boost mini to 8 MIPS.
E Basart, D. folger, B. Shellooe (Ridge).
Mini-Micro Systems sept. 82 p. 258-272.

BAT82 L'après CP/M: OS9, un système d'exploitation multi-tâches.
L. Baticle.
Micro-Systèmes sept.-oct. 82, p. 107.

BAU83 Comparison of MVS/SP version 1 release 1.3 (MVS/370) to MVS/SP version 2 release 1.0 (MVS/XA) on the IBM 3081K.
J F. Bauer, P. R. Dorn, R J Wicks (IBM).
IBM Technical Bulletin GG22-9325-00, juin 83.

BAU85 An MVS SRM Discussion.
T.F. Bauer, B R. Pierce.
IBM Technical Bulletin GG66-0201-00 avril 85.

BEE84 Amdahl announces dual operating system option.
J. Beeler.
Computerworld 3 déc. 84 p. 1.

BEN84 Supermicros pushing beyond original small business.
E. Bender.
Computerworld 21 mai 84 p. 10.

BER78 The IBM S/38: addressing and authorization.
V. Berstis, C.D. Truxal, J.G. Ranweiler.
IBM S/38 Technical Developments, 1978, p. 55-58.

BER81 Paging in MVS/SP with extended swap.
T. Beretvas, J. Sly.
IBM Technical Bulletin GG22-9206-01, oct. 81.

BER8I Le système de fichiers modulaires de Prologue.
P. Bertrand.
Minis et Micros numéro 205 p. 47-51.

BIB81 MVS/SP release 3 ASM considerations.
R.P. Bibolet, B.E. Kash.
IBM Technical Bulletin GG22-9253-00 août 81.

BIR73 Functional structure of IBM Virtual Storage Operating Systems: architecture and design of DOS/VS.
J.P. Birch.
IBM System Journal, numéro 4, 1973, p. 401.

BOT84 System design combines two popular Unix versions in one operating system.
R. Bott (Pyramid).
Computer Systems Equipment Design nov. 84 p. 39-44.

BOU85 Aspects négatifs d'Unix en application de gestion.
J.P. Bouhot.
Convention Informatique sept. 85, p. 237-238.

BOZ84 Analysis of free-storage algorithms revisited.
G. Bozman, W. Buco, T.P. Daly, W.H. Tetzlaff.
IBM System Journal vol 23 numéro 1, 1984 p. 44-64.

BRA79 Computing and Communications - A perspective of the evolving environment.
L.M. Branscomb.
IBM System Journal vol 18 numéro 2, 1979 p. 189-201.

BRO83A Tweating more performance from an Operating System.
D.K. Brown, K. Strutynski, J.H. Wharton.
Computer Design mai 1983, p. 193.

BRO84 Sceptre: proposition de noyau normalisé pour les exécutifs temps réel.
F. Browaeys, H. Derriennic, P. Desclaud, H. Falloir, C. Faulle, J. febvre, J.E. Hanne, M. Kronental, J.J. Simon, D. Vojnovic.
TSI vol 3 numéro 1, 1984 p. 45-62.

BRY85 User-friendly interfaces in the Info Centre.
D. Bryant (IBM).
SEAS proceedings 23-27 sept. 85, p. 1101-1108.

BUL75 GCOS Analysis.
Bull course code T50, 1975.

BUL77 Introduction to Level 64: R01 Student Book.
46A56503 Rev 1, Bull, oct. 77.

BUL77A Level 66 FMS analysis student handbook.
Bull, course code T90, 1977.

BUL78 GCOS6 data file organisations and formats.
Bull 69A2CB05 rev 0, janv. 78.

BUL80A MINI6 GCOS6-MOD400 system concepts.
Bull 69A2CB20 Rev 1, fév. 80.

BUN76 Scheduling Techniques for OS.
R.B. Bunt.
Computer vol 9 oct. 76 p. 10-17.

BUN85 Armonk: a portrait of success.
H. Bunn.
Computerworld Extra 4 déc. 85 p. 18-22.

BUR65 Le monde à l'heure des calculateurs.
G. Burk.
Dunod 1965.

BUZ78 A queueing network model of MVS.
J.P. Buzen.
ACM Computing Surveys, sept. 78, p. 319.

CAH85 La révolution lente des centres de calcul.
I - Automatisation complète: les conditions techniques sont déjà réunies.
J.P. Cahier.
Le Monde Informatique 9 sept. 85 p. 18-21.

CAS78 Architecture of the IBM System/370.
R.P. Case, A. Padegs.
Communications of ACM vol 21 numéro 1, janvier 78 p. 73-96.

CHA84 Présentation du système d'exécution répartie d'Intertechnique.
J.F. Champsaur.
Congrès sur les nouvelles architectures pour les communications sept. 84
Eyrolles, p. 243-250.

CHA86 Entretiens avec
C. Chassan.

CHE81A The UNIX operating system: portability a plus.
E. Cherlin.
Mini-Micro Systèmes avril 81, p. 153.

CHE85 ISPF: this dialog manager is hard to talk with.
D. Chen.
Computerworld Extra 4 déc. 85 p. 83-86.

CII76 Series 60 Level 66: system software overview.
66A2-DE rev 0.
CII-HB juillet 76.

CII78 Series 60 Level 64 GCOS. Job Control Language Reference Manual.
46A2AQ10 rev 3.
CII-HB mai 1978.

CLA78 MVS LPA optimization.
W.K. Clawson.
CMG Management & Evaluation of computer center, San Francisco, déc. 78.

CLA85 Performance of the VAX-11/780 translation buffer: simulation and measurement.
D.W. Clark, J.S. Emer.
ACM Transactions on Computer Systems vol 3 numéro 1, fév. 85 p. 31-62.

CLE86 The HP-UX operating system on HP precision architecture computers.
F.W. Clegg, G. Shiu-Fan Ho, S.R. Kusmer, J.R. Sontag.
Hewlett-Packard Journal déc. 86, p. 4-22.

CLE8X Systèmes d'exploitation pour micro-ordinateurs: Memdos et ses catactéristiques.
P. Clerc.
Minis et Micros numéro 194, p. 93-99.

COI84 Systèmes et langages orientés objets.
P. Cointe (Ircam).
Convention Informatique 84, p. 87-98.

COL82 An object oriented OS for microcomputers.
J.P. Collins, M.M. Lewitt.
Computer Design, juin 1982.

CON86 Contourner les mots de passe.
W. Connolly.
L'informatique Professionnelle num 44, mai 86 p. 27-36.

COO84 Pick steps out.
R.Cook.
Computerworld 9 avril 84 p. ID/1-8.

COR81 Systèmes informatiques répartis.
CORNAFION.
Dunod 1981.

CRA81 Trends in DP budgets.
J. Crane.
Datamation p. 140-150.

CRI82 Transforming an OS.
P. Crisman, G.E. Bryan.
Datamation mars 82, p. 158.

CRO75 Systèmes d'Exploitation des ordinateurs.
CROCUS.
Dunod, 1975.

CW83 Fault-tolerant banking system 68000 based (Auragen).
Computerworld, 18 avril 1984, p. 68.

CW85A Piracy.
Computerworld 21 janv. 85 p. 4.

CW86A Distribution of OS on mainframes.
Computerworld 10 mars 86 p. 25.

DAH78 The IBM System/38: a high levl machine.
S.H. Dalby, G.G. Henry, D.N. Reynolds, P.T. Taylor.
IBM S/38 Technical Developments 1978 p. 47-50.

DAH84 Les systèmes d'exploitation des micro-ordinateurs.
M. Dahmke.
McGRAW-HILL ISBN 2-7042-1071-3, 1984.

DAL84 Software portability issues confront computer OEMs.
R. Dalrymple.
Mini-Micro Systems juin 84 p. 113-116.

DAT82 In focus: Gary Kildall on CP/M.
Datamation octobre 82, p. 39.

DAL84A Distributed process control architectures meet plant data needs.
R. Dallimonti (Honeywell).
Computer technology review summer 84 p. 351-360.

DEC82 VAX/VMS System Management and Operations Guide.
DEC AA-M547A-TE, mai 1982.

DEN84 Les systèmes d'exploitation.
Y. Denning, R. Brown.
Pour la Science, novembre 1984, pages 52-61.

DIE85 Workload configuration in a JES3 complex.
W. Diefenbach.
SEAS proceedings 23-27 sept., p. 105-121.

DJU85 Up the software curve.
B. Djurdjevic.
Datamation 15 mai 85, p. 96.

DOH79 Managing VM/CMS Systems for user effectiveness.
W.J. Doherty, R.P. Kelisky.
IBM Systems Journal numéro 1, 1979, p. 143.

DUB85 Unix comes in many flavors. It's all a matter of taste.
M. Dubrall.
Computerworld Extra 16 janv. 85 p. 45-48.

DUB86 Conversations avec
J.M. Dubois.

DUB8X Tour d'horizon des systèmes d'exploitation: conséquences et perspectives.
R. Dubois.
Minis et Micros numéro 212 p. 28-32

ELK86 Upgrading proves cost-effective for immediate PC expansion.
T.A. Elkins.
Computerworld 5 mai 86 p. 65-80

EME84 The Unix universe.
S. Emerson.
Datamation août 85 p.76.

EMM8I VM/CMS on a mini.
R. Emmet, J. Verity.
Datamation p. 82.

ENG75 Capability Concept mechanism and structure in System 250.
D.M. England.
RAIRO septembre 75 p. 46.

ENS78 What is a "Distributed" Data Processing System ?
Ph. Enslow.
Computer vol 11 juin 1978 p. 13-21.

FAJ85 A Unix OS adapted for a technical Personal Computer.
R.M. Fajardo, A.L. Rood, J.R. Andreas, R.C. Cline.
HP Journal oct. 85, p.22.

FAR77 Converting to MVS.
K.P. Farrell.
Datamation fév. 77, p. 121.

FON8I Systèmes d'exploitation: le concept "temps réel" dans l'environnement iRMX86.
E/S et protection des fichiers.
A.B. Fontaine.
Minis et Micros numéro 197 p. 51-55; numéro 198 p. 87-91.

FRA79 Logiciel: nouveaux aspects économiques.
W.L. Frank.
L'informatique nouvelle oct. 79 p.24-29.

FRE82 Portable operating systems fight for 16-bit machines.
D. Freedman.
Mini-Micro Systems sept. 82 p. 237-249.

FRE84 Silicon Valley.
P. Freiberger, M. Swaine
McGraw-Hill, ISBN 2-7042-1105-1, 1984.

FRI81 DASD Access Methods performance considerations.
S.E. Friesenborg (IBM).
IBM Technical Bulletin GG22 9241 juillet 81.

FRI81A MVS paging performance considerations.
S.E. Friesenberg (IBM).
IBM Technical Bulletin GG22-9264-00, nov. 81.

FRI85 MVS performance managemnt.
S.E. Friesenborg, G.L. Hall.
IBM Technical Bulletin GG22-9351-00 janv. 85.

FRY Software Portability
C H Frye
Computerworld, In Depth.

GEM85 Automating machine room operation.
P. Gemis (IBM).
SEAS proceedings 23-27 sept., p. 135-149.

GIE85 Unix: standard de système d'exploitation(II).
M. Gien.
Ø1 Informatique numéro 194 nov. 85 p. 57

GIL86 Pick operating system makes converts of users.
Ph. G...
Computerworld, In Depth, 24 mars 86, p. 89-101.

GIS8X Le p-System: représentation des données et adaptation à une configuration d'accueil.
Le p-System: un système d'exploitation conçu pour la portabilité.
Ch. Gissinger, J.M. Pironneau.
Minis et Micros numéro 214 p. 29-34; numéro 212 p. 57-61.

GLA85 VM/SP4 user experience.
I. Glass (IBM).
SEAS proceedings 23-27 sept. 85, p. 349-375.

GRO84 La mémoire virtuelle et le MC 68010.
P.F. Groepler (Motorola).
Electronique Industrielle numéro 75, 15 sept. 84.

GRO86 Des systèmes d'exploitation sur disque ou sur silicium qui gèrent le temps réel.
J. Gross
Electronique Industrielle numéro 104, 15 mars 1986, p. 74-80.

GUI82 The Chorus distributed operating system: design and implementation.
M. Guillemenont (INRIA).
Local Computer Networks, North-Holland, 1982, p. 207.

GUI84 Etude comparative de quelques systèmes répartis.
M. Guillemont (Inria).
TSI vol 3 numéro 1, 1984 p. 5-21.

GUM83 System/370 Extended Architecture: facilities for Virtual Machines.
P.H. Gum.
IBM Journal Research and Development, vol 27, numéro 6, novembre 1983, p. 530.

HAB76 Modularization and hierarchy in a family of operating systems.
A.N. Habermann, L. Flon, L. Cooprider.
Communications of the ACM, mai 76, p. 266.

HAM83 Implantation de Xenix sur microprocesseur 16 bits.
PH. Hammes (Intel).
Electronique Industrielle 15 oct. 83 p. 71-74.

HAR82 Workload characterization and standardisation.
P. Hartmann.
ECOMA proceedings oct. 82, p. 104.

HAR82A Construction of synthetic models for workload characterization.
G. Haring.
ECOMA proceedings oct. 82, p. 235.

HAR83 Interface logicielle pour les familles iAPX 88 et 86: vers la standardisation.
A.C. Hartmann.
Minis et Micros numéro 184, p. 39.

HAR84 Upgrading to multi-user.
J. Hartley.
Systems International déc. 84 p. 82.

HAR84A The micro OS gap.
N. Harrison.
Systems International déc. 84 p. 65.

HAR85 PC Network outperforms Unix.
C.A. Harshman.
Mini-Micro Systems mars 85 p. 93.

HEL75 Computer System Performance.
H. Hellerman, Th.F. Conroy.
McGRAW-Hill ISBN 0-07-027953-5, 1975.

HEN79 Evolution of a virtual machine subsystem.
E.C. Hendricks & T.C. Hartmann.
IBM Systems Journal numéro 1,1979, p. 111.

HES84 IMS/DB2: IBM's data base strategy for tomorrow.
P.E. Hessinger.
Computerworld 9 avril 84, p. ID/9-30.

HEY84 The nonsense of Unix.
T. Heywood.
Systems International avril 84 p. 91.

HIN84 Benchmarking UNIX systems.
D.F. Hinnant.
Byte août 84 p. 132.

HOA82 UNIX more standardized than CP/M, runs with more packages.
B. Hoard.
Computerworld, 15 fév. 82, p. 67.

HO84 DMERT - An operating System for telecommunications systems.
S.F. Ho, W.C. Schwartz (AT&T).
AFIPS Conference Proceedings NCC vol 53, 1984 p. 23-29.

HOL78 Planification du matériel et détermination de la capacité.
J.R. Holt (BCS).
Présentation au LMI, 1978.

HOL83 MP/M-86 handles real-time and multi-user business applications.
F. Hilsworth (DRI).
Mini-Micro Systems fév. 83 p. 253-258.

HUC84 CP/M meets Unix.
T. Huckvale.
Systems International avril 84 p. 94.

HUR85 Managing TSO: a measurement approach.
P. Hurst (Morino Ass.).
SEAS proceedings 23-27 sept. 85, p.539-570.

IBM78 OS/VS2 MVS overview.
IBM GC28-0984-0, juin 78.

IBM79 DOS/VSE System Control Statement.
IBM GC33-5376-6, fév. 1979.

IBM79A VSE/Adavanced Functions, macro user's guide.
IBM SC24-5210-0, oct. 79.

IBM80C OS/VS2 MVS overview.
IBM GC28-0984-1 mai 1980.

IBM82A VM/SP operator's guide.
IBM SC19-6202-1, avr. 82.

IBMPE OS/VS2 MVS performance notebook.
IBM GC28-0886.

IBMXX OS/VS2 MVS Resource Management Facility (RMF), Reference User's Guide.
IBM SC28-0922-4.

INR80 Chorus: an architecture for distributed systems.
INRIA, rapport numéro 42, novembre 80, 67 pages.

INS85 An approach to successful online transation processing application.
A. Inselberg (Synapse Computer Corp.).
AFIPS Conference Proceedings NCC vol 54, 1985 p. 419-427.

INT83 Introduction to the system 86/380 microcomputer system.
Intel 172758-001 mars 83.

INT83A Introduction to the iRMX86 operating system.
Intel 9803124-04 1983.

INT83B iRMX86 programmer's reference manual Part I for release 6.
Intel 146195-001 1983.

INT85D iAPX286/10 high performance microprocessor with memory management and protection.
Intel 210253-008, mai 1985.

INT85G 80386 high performance microprocessor with integrated MMU.
Intel 231630-001 oct. 85.

IRW81 Multitasking executive speeds 16 bit micros.
J.M. Irwin
Electronic Design, vol 29 numéro 5, mai 81.

JES80C OS/VS2 MVS System Programming Library, JES3 Release 3.
GC28-0608-6 mars 1980.

JES83A MVS/XA JES3 introduction.
G23-0049-00, IBM janv. 83.

JES83B MVS/XA JES3 commands.
SC23-0063-0, IBM janv. 83.

JES3C MVS/XA JES3 overview.
SC23-0050-0, IBM janv. 83.

JON75 Towards design of secure systems.
A.K. Jones, W.A. Wulf.
Software Practice and Experience vol 5 num 4 oct-déc. 75 p. 321-336.

JON85 Single-board computers can tap each other's multiprocessor OS.
K. Jones.
Mini-Micro Systems janv. 85 p. E2.

JOU84 UNIX "LE" système d'exploitation.
P. Jouvelot, D. Leconte des Floris.
Micro-systèmes mai 84, p. 90-98.

KAD86 What's ahead for CICS.
J. Kador.
Datamation 1 sept. 86, p. 77-80.

KAH85 What is a user-friendly program?.
B. C. Kahan (IBM).
SEAS proceedings 23-27 sept., p. 277-286.

KAI84 Problèmes actuels des systèmes répartis.
Cl. Kaiser (CNAM).
Congrès sur les nouvelles architectures pour les communications sept. 84, p. 251-258.

KAL80 Operating systems comparison.
F. Kaltseis.
ECOMA newsletter, mars 80, article numéro 9.

KEA86 Des performances à la hauteur des applications de pointe.
T.J. Keating, J.N.L. Prak, K. Schoemaker (Intel).
Electronique Industrielle numéro 103, mars 86 p. 31.

KEE77 An outline of the ICL 3900 series system architecture.
J.L. Keedy.
Australian Computer Journal vol 9 num 2, juillet 77, p. 53-62.

KEN82 Database software package for micros.
P. Kenealy.
Mini-Micro Systems sept. 82 p. 193-202.

KIN83 Centralised versus decentralised computing: oganisational considerations and management options.
J.L. King.
Computing Surveys vol 15 numéro 4, déc. 83, pages 319-349.

KIN84 Virtual memory system provides added density within a limited area.
R.A. King (Motorola).
Computing Technology Review summer 84 p. 327.

KIN85 MVS/SP2.1.2 ASM enhancements. G.M. King (IBM).
SEAS proceedings 23-27 sept. 85, p. 1169-1207.

KNI82 Capacity planning: an historical perspective.
A. Knight.
ECOMA proceedings oct. 82, p. 122.

KOE85 VM response time measures in VMMAP, VMPPF and VMRTM.
K.W.P. Koeppel.
SEAS Proceedings sept. 85, p.73-86.

KOK82 Analisis and application of an IMS analytical model.
M. Kok.
ECOMA proceedings oct. 82, p. 171.

KOL82 Tuning a high performance transaction processing system.
G. Kolenaty.
ECOMA proceedings oct. 82, p. 191.

KON85 A view on windows: current approaches and neglected opprotunities.
B.R. Konsynski, A. Greenfield, W.E. Bracker.
AFIPS Conference Proceedings NCC vol 54, 1985 p. 451-460.

KUT8X Wither VM?
D. Kutnick.
Datamation p. 73.

LAG81 Le système MT25.
J.P. Lager.
Commutation & Tansmission numéro 3, 1981 p. 81-98.

LAG85 Des systèmes de développement pour microprocesseurs distribués et centralisés.
G. Lagrange.
Electronique Industrielle numéro 81 15 janv. 85 p. 63-69.

LAM76 Reflexions on an OS design.
B.W. Lampson, H.E. Sturgis (Xerox).
Communications of the ACM, mai 76, p. 251.

LAM77 Performances CICS.
Lamothe (EDF).
AFCET, réunion du groupe de mesures du 12 janv. 1977.

LAU82 Information manager.
P. Laurie.
Systems International oct. 82 p. 61.

LAU84 The multi-user misconception.
P. Laurie.
Systems International déc. 84 p. 62.

LAW80 Understanding Computer systems.
H.W. Lawson.
Computer Age, déc. 80 p. 15-18.

LEC85 Les forces de rupture ou les défis de l'informatique des années 85-90.
J.Lechaczynski (IBM).
Convention Informatique sept 85, p. 481-487.

LEE84 Decision support in a distributed environment.
D.T. Lee (Univ. of Hartford).
AFIPS Conference Proceedings NCC vol 53, 1984 p. 477-488.

LEI85 JES3 performance tuning.
C.N. Lei (IBM).
SEAS proceedings 23-27 sept. 85, p. 793-805.

LEW76 Service levels: a concept for the user and the computer center.
L.J. Lewis.
IBM Systems Journal numéro 4, 1976, p. 328.

LEW82 Unix and the supermicrocomputer: a marriage of convenience.
G. Lewis.
Mini-Micro Systems nov. 82 p. 17-22.

LEW82A Pick jockeys for position in standard operating system race.
G. Lewis.
Mini-Micro Systems sept. 82 p. 32-42.

LIS79 Primitives for distributed computing.
B. Liskov.
7-th Symposium on OS principles, Pacific grove, California, déc 79.

LIS83 Principes fondamentaux des systèmes d'exploitation.
A.M. Lister.
Eyrolles, 3e édition, 1983.

LMI86 Logiciels IBM: les prévisions de Dale Kucknick.
Le Monde Informatique 14 avril 1986, p. 10.

LMI86A COCS: les utilisateurs réservés sur la version 1.6.
Le Monde Informatique 14 avril 1986, p. 54-55.

LOP84 Access control lists in capability environments.
L. Lopriore.
TSI vol 3 numéro 3, 1984 p. 179-191.

LS85A La stratégie Burroughs au travers de MCP.
Logiciels & Services numéro 46, avril 85 p. 34-37.

LS86A Micro: les SGBD «relationnels».
Logiciels & Services févr. 86 p. 43-47.

LS86B Selon le rapport de l'ADI: les chiffres de l'informatique.
Logiciels & Services févr. 86 p. 40-42.

LS86C Réseau: l'architecture Domain.
Logiciels & Services févr. 86 p. 34-35.

LUK86 HP precision architecture performance analysis.
J.A. Lukes.
Hewlett-Packard Journal août 86, p. 30-39.

LUN79 Forecast for the '80s IBM Control Systems with more functions.
E.D. Lundell.
Computerworld 14 fév. 1979 p. 27.

LUN82 MVS/370 Tightly Coupled Concepts.
S. Lunn, R. Wicks.
IBM Technical Bulletin GG22-9276-01 mai 1982.

LUN85 An introduction to task management in the iRMX86 OS.
C.J. Lundberg (Intel).
Solutions juillet/août 85 p. 24.

MAD85 How multiprocessor nodes can become more sociable.
K. Madsen, D. Foley (Tandem).
Data Communications sept. 85, p. 279-294.

MAI85 Architecture du logiciel d'un autocommutateur construit sur un réseau de microprocesseur.
M. Maisonneuve, JP. Levy, JL. Konrat.
Commutation & Tansmission.

MAR82B Talks about the data base environment.
J. Martin (interview).
Computerworld, In Depth, 4 oct. 82.

MAR82A An overall plan.
J. Martin.
Computerworld, In Depth, 4 oct. 82.

MAR84 Unix: the search for a standard.
R. Marsh.
Computerworld Extra 26 sept. 84, p. 29.

MAR84A Problèmes posés par le développement de logiciels réseau sous Unix.
B. Martin, H. Lucas, V. Merrien (CNAM).
Congrès sur les nouvelles architectures pour les communications sept 84, p. 235-241.

MAR86 Peaceful coexistence: the Unix versus MS-DOS debate.
R. Marsh.
Computerworld 17 mars 86 p. 17.

MAR86A High user overhead mars interface design.
N. Margolis.
Computerworld 28 avril 86, p. 65-72.

MAT85 RACF under VM.
D.R. Matthews.
SEAS proceedings 23-27 sept. 85, p.741-759.

MAU85 MVS/System Product - JES2.
R.A. Mauri, L.J. Orza (IBM).
SEAS proceedings 23-27 sept. 85, p. 761-791.

MEL82 Four DBMS trends to take Industry in 90's.
T. Melchiorre.
Computerworld, 25 oct. 82, Special Report p. 9.

MEN82 Modelling of CPU preemptive resource priority.
D. Menascé.
ECOMA proceedings oct. 82, p. 208.

MIM82 Bug-free systems.
P.Mimno.
Computerworld 11 oct. 82, In Depth.

MIN8X Artémis et Soprano: deux exécutifs en temps réel pour microprocesseurs 16 et 32 bits.
Minis et Micros numéro 211 p. 71-75.

MKI79 The changing virtual machine environment: interfaces to real hardware, virtual hardware, and other virtual machines.
R.A. MacKinnon.
IBM Systems Journal numéro 1, 1979, p. 18.

MLE84 Alternative operating systems.
M. McLening.
Systems International juin 84 p. 79.

MLE84A Why System V?
M. McLening.
Systems International mai 84 p. 89.

MMS85 Operating systems directory.
Mini-Micro Systems fév. 85 p. 164-175.

MMS85A What is PC-AT compatibility?
Mini-Micro Systems mars 85 p. 36-43.

MOA86 Orphan in a storm.
J. Moad.
Datamation 1er sept 86', p. 22-24.

MON80 Measurement of a 4331 with a hardware monitor.
J.A. Montagnon.
Ecoma Newsletter, 1980.

MOR84 How not to worry about Unix.
D. Morris.
Datamation août 84 p. 83-86.

MOR85 The evolution of software architecture.
D.V. Morland.
Datamation fév. 85 p.123-132.

MOS83 Decision Support System (DSS) analysis.
R. Moskowitz.
Interface Age, avril 1983, p. 62.

MOY84 DOS/VSE: what VSE/SP 2.1-0 means for users.
B.I. Moyle.
Computerworld, 30 avril 84, In Depth, pages 49-51.

MUD85 An information technology architecture for change.
M.W. Mudie, D.J. Schafer.
IBM Systems Journal vol 24 num. 2/3 1985 p. 307-315.

MUL82 Honeywell DPS 8/Multics. Summary description.
Honeywell DL92-01, 1982.

MUL83A Operator's guide to Multics.
Bull GB61-00, 1983.

MYE82 UNIX: a standard now?
E. Myers.
Datamation, janvier 82, p. 77.

NAS85 Single board solution technical reference.
NAS FE-05GW202-0 avril 85.

NBS78 The system management facilities (SMF) for IBM systems under OS/MVT.
NBS special publication 500-40, oct. 78.

NCP83 ACF/NCP for the IBM 3725 diagnostic reference.
IBM LY30-3071-0 août 83.

NEU78 Computer security evaluation.
P.C. Neumann.
AFIPS Conference Proceedings vol 47 NCC, p. 1087-1095.

NIC81 Communication through message passing or shared memory: a formal comparison.
R. De Nicola, A. Martelli, U. Montanari.

NIE85 Errors in dialog design and how to avoid them.
J. Nievergelt.
SEAS proceedings 23-27 sept., p. 247-256.

NOR80 Description du logiciel Norsk Data.
Norsk Data.

NOR81 The trouble with UNIX.
D.A. Norman.
Datamation nov. 81, p. 139.

ORD86 Si tu ne viens pas à XA...
Ordinateurs 23 juin 86, p. 14-16.

ORD86A 4381: vendre ou acheter.
Ordinateurs 14 mai 86, p. 12-18.

OUA8X Successeur du Z80, le Z8108 est le premier membre de la famille Z800.
M. Ouaknine.
Minis et Micros numéro 185 p. 35-38.

OUS79 Medusa: an experiment in distributed OS structure.
J.K. Ousterhout, D.A. Scelza, P.S. Sindhu.
7-th Symposium on OS principles, Pacific Grove, California, déc 79.

OVE82 Implementing the UCSD p-System.
M. Overgaard.
Mini-Micro Systems sept. 82 p. 183-190.

PAA82 Analyzing CPU usage by MVS and application routines.
R. Paans.
ECOMA proceedings oct. 82, p. 94.

PAR77 Analyse et mesure des systèmes de fichiers.
Ch. Parent.
RAIRO vol 11 numéro 3 1977, p. 235-254.

PAR81 Adapting program behavior to virtual memory environments.
J.F. Paris.
Convention Informatique 1981, p. 237-251.

PAT83 An inside look at MS-DOS.
T. Paterson.
Byte juin 83, p. 230.

PAT85 Z8000 32-bit microprocessor.
A. Patel.
AFIPS Conference Proceedings NCC vol 54, 1985 p. 225-231.

PAU82 Continued rise in MVS use among five trends spotted at IBM maonframe sites.
L. Paul.
Computerworld 5 avril 82, p. 10.

PER8X L'évolution des outils de mise au point des systèmes à microprocesseurs.
H. Perceval.
Minis et Micros numéro 194 p. 119-124.

PET78 Simulating Working Set under MVS.
A. Petrella, H. Farrey.
CMG Management & Evaluation of Computer Center, San Francisco, déc 78.

PHI86 Sophonet System Manual.
Philips 5122 991 34681, mai 86.

PHI86A Distributed Real time Multiprocessor System PG5000. System Concepts.
Philips 9499 060 22011 840712, 1986.

PIG86 Users tame CICS.
S. Piggot.
Computerworld In Depth, 19 mai 1986 p. 77-81.

PIN78 The IBM S/38: object oriented architecture.
K.W. Pinnow, J.G. Ranweiler, J.F. Miller.
IBM S/38 Technical Developments, 1978 p. 51-54.

PIN86 Discussions diverses avec
A. Pinet.

POP85 Make friends with Unix.
S. Pope.
Systems International fév. 85 p. 73.

POU86 Tripos, the roots of Amigados.
D. Pountain.
Byte fév. 86 p. 321

POW85 PC clones tests.
N. Powell.
Systems International juin 85 p. 41.

PRI82 UNIX: le système de gestion de fichiers et les volumes montés.
V. Prince.
Minis et Micros 18 oct. 82, numéro 173, p. 71.

PRI82B UNIX: les entrées-sorties standard et les «tubes» de communication.
V. Prince.
Minis et Micros 15 nov. 82, numéro 175, p. 59.

RAS84 Accent: a distributed operating system kernel for a network of personal computers.
R.F. Rashid.
Convention Informatique 84, p. 77-86.

RAU86 Real-time Unix seizes new products, markets.
W. Rauch-Hindin.
Mini-Micro Systems sept. 86, p. 61-78.

REI73 Computer Decisions.

RET84 VAX clusters.
T. Retford (DEC).
Systems International mars 84 p. 63.

RIF82 UNIX=Unique.
C. Rifaux.
L'Informatique Professionnelle numéro 5, août-sept 82, p. 132.

RIF84 "You want Unix? We've got Unix!".
G. Rifkin.
Computerworld Extra 26 sept. 84, p. 11.

RIT74 The UNIX Time-sharing System.
D.M. Ritchie, K. Thompson (Bell Laboratories).
Communications of the ACM janv. 83, p. 83-89.

ROG85 JES2 and JES3 - A functional comparison.
P. Rogers (IBM).
SEAS proceedings, 23-27 sept. 85, p.93-103.

ROL85 Processor board handles virtual memory.
J.J. Roloff.
Mini-Micro Systems mars 85 p. 173.

ROU78 Mots de synchronisation.
G. P. Roucairol.
RAIRO vol 12 numéro 4, 1978 p. 277-290.

ROU81 VM est-il stratégique pour IBM?
F. Roux.
Le Monde de l'Informatique 9 nov. 81, p. 8.

RTE83C RTE-A Programmer's Reference Manual.
Hewlett-Packard 92077-90007 juin 83.

RTE83D RTE-A User's Manual.
Hewlett-Packard 92077-90002 juin 83.

RYA85 More of "All about VM paging".
P. Ryall (IBM).
SEAS proceedings 23-27 sept. 85, p. 1059-1075.

SAL75 The protection of information in computer systems.
J.D. Saltzer, M.D. Schroeder.
Proceedings of IEEE vol 63 num 9 mars 75, p. 1278-1308.

SAL86 Implantation d'un moniteur temps réel multitâches s'exécutant sur une carte d'extension de l'IBM PC.
D. Salember.
Electronique Industrielle numéro 104 mars 86 p.49.

SAL86A PC-Unix links penetrate office.
D. de Salvo.
Mini-Micro Systems janv. 86 p. 83.

SAU83 UNIX - a power for the present.
D. Saunderson.
i fév. 83 p. 46.

SCH81 More on UNIX.
G. Schmeder.
Datamation, déc 81, p. 24.

SCH81A The iRMX 86 OS.
B. Schafer, J. Davis (Intel).
Intel OE Systems Handbook, 1983, pages 5-294.

SCH84 Toward an understanding of extensible architectures for evaluated trusted computer system products.
M. Schaefer, R.R. Schell (DOD).
Proceedings 1984 Symposium on Security and Privacy, p. 41-49.

SCH85 MVS under VM performance considerations.
A. Scher
IBM Technical Bulletin GG22-9398-00 fév. 85.

SEA79 VM/370 - a study of multiplicity and usefulness.
L.H. Seawright, R.A MacKinnon.
IBM Systems Journal numéro 1, 1979, p. 4.

SHA77 The total computer security problem.
K.S. Shankar.
Computer juin 77 p. 50-73.

SHE73 Functional structure of IBM Virtual Storage Operating System: OS/VS2-2 concepts and philosophies.
A.L. Sherr.
IBM System Journal numéro 4, 1973, p. 382.

SHI85 IBM PC family BIOS comparison.
J. Shiell, J. Markoff.
Byte fall 85, p. 259-265.

SI82 Design considerations for a microcomputer task monitor.
Systems International oct. 82 p. 67.

SIE82 The theory and practice of reliable system design.
D.P. Siewiorek, R.S Swarz.
Digital Press EY-AX016-DP ISBN 0-932376-13-4.

SIG84 UNIX's infancy.
L. Sigler.
Computerworld 26 sept. 84, p. 75.

SIL84 The Unix system guide book.
P.P. Silvester.
Springer Verlag ISBN 0-387-90906-0.

SIM84 Proteus, une architecture de système distribué.
F. Simon.
Convention Informatique 84 p. 99-102.

SIM86 Multiprocessors use radical architectures.
D. Simpson.
Mini-Micro Systems mai 86 p. 7787.

SLA85 User friendly systems.
J.B. Slater.
SEAS proceedings 23-27 sept. 85, p. 621-638.

SMI82 Why to go to MVS/XA.
J.P. Smith, R.J. Wicks.
IBM Technical Bulletin GG22-9302-00 déc 82.

SMT82 Manuel de référence FLEX-9, Manuel de prgrammation avancée FLEX-9 Goupil.
SMT octobre 1982.

SOU84 Un exécutif temps réel pour les MicroVax.
B. Soukoff (DEC).
Electronique Industrielle 15 nov. 84 p. 82-87.

SRE80 An experimental study of relative throughput in a multiprocessor computer system.
K. Sreenivasan, G.A. Nelson, J.A. Maksin.
Software Practical & Experience vol 10, 1980, p. 973.

SUM84 An overview of computer security.
R.C. Summers.
IBM System Journal vol 23 numéro 4, p. 309-325.

TAN75 Non Stop systems introduction.
Tandem T16/8003 82003, 1975.

TAN82 Safeguarding today's interactive computer systems.
J.D. Tangney, P.S. Tasker.
Computing Security Journal vol 1 num 22 winter 82, p. 57-70.

TAN83 The design of a capability-based distributed operating system.
A.S. Tanenbaum, S.J. Mullender.

TAN86 Concepts and facilities.
Tandem 38401-A00-00 1986.

TAY85 A Pick Defence.
M. Taylor.
Systems International août 85 p. 61.

TEL84 Windows on Unix.
E. Tello.
PC Magazine 12 juin 84 p. 228-239.

TET79 State sampling of interactive VM/370 users.
W.H. Tetzlaff.
IBM Systems Journal numéro 1, 1979, p 161.

TET84 VM/370, attached processor and multiprocessor performance study.
W.H. Tetzlaff, W.M. Buco.
IBM System Journal vol 23 numéro 4, 1984 p. 375-385.

THO82 The UNIX system.
R.A. Thomas, J. Yates.
Computerworld, In Depth.

THO85 Le marché de la micro-informatique dans la dernière ligne droite.
C. Thomas.
Convention Informatique sept 85 p. 43-46.

TOM85 VM/XA systems facility ESP, performance, and RTM/SF.
J.D. Tomaselli (IBM).
SEAS proceedings 23-27 sept. 85, p.1157-1168.

TRU82 Generic Operating Systems: bringing the pieces together in office systems.
P. Truax.
Computerworld In Depth 29 nov. 82.

TUC85 Unix emerges as a universal tool kit.
M. Tucker.
Mini-Micro Systems fév. 85 p. 149-160.

TUC86 32-bit chips reshape Unix industry.
M. Tucker.
Mini-Micro Systems janv. 86 p. 69.

TUC86A The IBM 3090 system: an overview.
S.G. tucker.
IBM System Journal vol 25 num 1, 1986 p. 4-19.

TUR76 Privacy and security issues in information systems.
R. Turn,W.H. Ware.
IEEE Transactions on Computers vol C-25 num 12 déc 76, p. 1353-1361.

UNG86 The ITT XTRA XP.
J.D. Unger.
Byte juillet 86 p. 281-290.

VAL84 Hardware differences can make or break the 50 IBM PC-compatible manufacturers.
L. Valigra.
Mini-Micro Systems juillet 84 p. 97-104.

VAX82 Structures du VAX (VAX concepts), guide de l'étudiant.
DEC EY-1102F-SG-0001 juin 82.

VAX82A VAX/VMS Primer.
DEC AA-D030C-TE mai 82.

VAX84A Guide to VAX/VMS system security.
DEC AA-Y510A-TE (VMS 4.0).

VAX84B Guide to VAX/VMS file applications.
DEC AA-Y508A-FE sept. 84.
VER85 1985 DP budget survey.
J.M. Verity.
Datamation 15 mars 85 p. 74.
VIA81 La nouvelle gamme de centraux temporels de CIT-Alcatel.
F. Viard, J.B. Jacob, P. Fritz
Commutation & Tansmission p. 87-102.
VIL84 Fault-tolerant Unix supports commercial transaction processing.
G. Villapiano (Auragen).
Computer Technology Review summer 84 p. 335-341.
VM79A VM/370 concepts et fonctions.
IBM SRF2-4589-0, 1979.
VMS84A VAX/VMS I/O user's reference manual Part II.
DEC AA-Z601A-TE sept. 84.
VMS84E Guide to VAX/VMS performance management.
DEC AA-Y515-TE, 1984.
VMS85A VAX/VMS I/O user's reference manual Part I.
DEC AA-Z600B-TE juil. 85.
VMS85B VAX/VMS system services reference manual.
DEC AA-Z501B-TE juil. 85.
VSE84 General Information, VSE/System Package Version 2 Release 1 Modification level 0
IBM GC33-6176-0 mars 84.
VUL78 Vulcan concepts and features.
Harris, 0862003-003, mai 78.
WAI85 Climbing around in the DOS directory tree.
M. Waite, J. Angermeyer, M. Noble.
PC Magazine 12 juin p. 275-288.
WAN80 VS System overview
Wang Laboraories Inc, 1980.
WAR85 Environmental software opens system windows.
C. Warren.
Mini-Micro Systems mars 85 p. 115-131.
WAT86 MS-DOS creator Tim Paterson earns place in industry annals.
P. Watt.
Computerworld 7 avril 86 p. 54-56.
WEB84 Opening windows.
0185A Tout, tout, tout... sur Sierra.
01 Hebdo 4 mars 85, p. 7-18.
WIC82 MVS/XA: an overview.
IBM Technical Bulletin GG22-9303-00 nov. 82.
WIT84 A portable Modula-2 operating system: SAM2S.
L.D. Wittie, A.J. Frank.
AFIPS Conference Proceedings NCC vol 53, 1984 p. 283-292.
WOH85 DASD error recovery under MVS/XA (user experience).
G. Wohlrab.
SEAS proceedings 23-27 sept. 85, p. 467-489.
WOL81 Microcomputer OS trends.
P.J. Wolochow.
Digital Design, déc 81.
WOO84 MVS/XA RMF Monitor III.
J. Woods.
IBM Technical Bulletin GG22-9350-00 janvier 84.
WRA8X Microprocesseur 32 bits à hautes performances: l'architecture du NS32032.
G.A. Wray.
Minis et Micros numéro 212 p. 63-67.
YEL85 CICS - An evolving system facility.
B.M. Yelavich.
IBM System Journal vol 24 numéro 3/4, 1985 p. 264-278.
ZAK81 Guide du CP/M avec MP/M.
R. Zaks.
Sybex, 1981.
ZIE79 A distributed information system study.
K. Ziegler.
IBM System Journal vol 18 numéro 3 1979, p. 374-401.
ZIM81 Basic concepts for the support of distributed systems: the CHORUS approach.
H. Zimmermann, JS Banino, A. Caristan, M. Guillemont, G. Morisset (INRIA).
2-nd International Conference on Distributed Computing systems, avril 81, Paris.

annexe E

index

abréviation 199 **254** 255
absolutisation 86 **174**
accès (voir *concurrence*, *méthode*, *mode*)
achat 19 26
acteur 24 25 147 236
administration 26 155 228 232
adresse 85 87 99 101 103 105 120 123 124 130-132 136 157 159 163 165 168 169 171-173 175 176 178 180 192 199 201 204 205 208 219 224 232 240 243 248 257 267
adresse physique/réelle 169 172 174-177 181 201 209 216 217 225 251
adresse virtuelle 172 174-176 180 181 201 204 208 216 217
agent 24 110 161 233
aide 153 154
alarmes 73 112 228 247
aléatoire 124 **127** 128 185
allocation 24 25 38 43 **46** **47** 48-50 52 53 55 **61** 63-69 71-74 75 79 81 82 95 98 104 109 **115** 117 119 124 141 146 152 153 156 163 164 166 189 191 213 214 224 237 271
Amigados 35 73 140
amorçage 69 71 73
analyse 37 **39** 40 41 43 44 48 53 135
anneau 194 206 215 218 234 236-243 245-247
antémémoire 95 98 146 151 168 173 **174**-177 184 197 200 201 204 218 236
anticipation 45 50 58 92
AOS 35 62
appareil 28 33
appareil dédié 52
appareil périphérique (voir *périphérique*)
appareil virtuel 51 **52** 96
application 15 16 18 26 33 34 38 43 **53** 55 56 62 64 81 83 87 93-95 97 109 110 119 125 126 136 140 141 143 144 146-149 151 156 158 164 172 181 221 224 226 227 229 234 253 267 269
arborescent 27 29 72 105 106 126 222 223
architecture **36** 75 107 144 145 152 156 158 163 174 201-203 218 223-230 232 250-252
arrière-plan **45** 99 184
article 39 75 83 **89** **90**-94 102 103 109 113 114 **118** 119-127128 129 161 267 270 271
AS 43 50 52 62 74 79 80 98 132 133 138 142 144-146 153 161 163 176 179 181 183 188 189 190-192 194-196 198 **201**-203 205 207 209-217 219 238 245-247 249-251
assistance 25
asynchrone 33 35 45 84 89 108 130 132 146 157 159 164 **220** 222 223
attribuer (voir *allocation*)
autorisation 20 24 26 30 40 44 50 54 64 100 103 106 **110** **111** 116 197 204 **207** 231-234 238-240 244-246 248 250 260
avant-plan **45** 271
bande 24 25 31 33 38 42 47 48 50 52 73 80 96 107 108 115 120 128 152 232 247 249 259 265 266
base de données 20 102 118 119 **124** **125**-127 143 144 147 149 155 156 161 227 269
bibliothèque 17 39 61 90 119 124 168 208 211-213 259 262 266
BIOS 57 89 100 135 214 226 246 265
bloc 27 67 69-71 73 83 **84** 89-92 95 96 98 99 103 109 117 120 123 **128** 129 132 158 159 170 171 183 184 189 221 236-238
boîte aux lettres 24 161 232
budget 21
cadre 48 74 **171** 172 174 176 179 183-190 192-195 197-199 205 210 211 213 216 235 237 238
calcul (voir *scientifique*)
canal 46 47 50 80 81 86 87 92 97-100 153 154 **174** 181 199 250 264 268
capacité 160 203 219 236 **244**-246
carte 25 30-33 38 39 47 48 50 52 72 92 96-98 134 137 138 152 220 252 259 265
case 163 164 171 173 181 184 188 195 236
cassette 47 107 120 247
catalogué 39 72 104 106 109 124 161 (voir *répertoire*)
CCITT 227 230
centralisé 45 55 68 69 71 72 74 105 106 218 160 164 170
charge 29 31 34 42 43 45 46 50 65 72 77 79 83 108 137 139 146 150 151 155 156 158 192 195 211 261 263 264 267-272 170 181

chargement 16 34 74 157 158 160 164 165 167-170 177-181 183-185 193 196 204 206 207 211-214 222 240 257-259 261 267
CHORUS 157 158
CICS 18 124 139 141 **144** **145**-147 149 224 261 272
classe **40** **41** 42 53
clé 69 120-123 127 129 132 138 151 154 164 194 196 202 211 213 214 232 236 237 254
CMS 28 29 35 113 114 132 133 140 146 **152** 153 155 253-255
collision 123 124
commande 28 31 **32** 37 40 42 48 50 53 61 62 66-68 72 73 79 81 91 95 97 98 103 106 107 109 110 113-115 128 132 135 136 140 142 143 189 190 197 208 211 214 215 223 232 233 253 254 255 256 258 262 264 271 (voir *JCL*, *OCL*)
communication 46 48 51 52 63 64 97 98 102 110 113 **130** **131** 133 139 144 148 150 152 156 159-161 201 210 212 220-222 224 **157**
commutation 35 81 144 155 157 165 172 173 176 202 205 220 230 241 242 262
compatible 18 19 36 66 87 151 203 224 239 240
compétition (voir *concurrence*)
compilation 16 17 21 42 62 90 148 168 267 268 271
compression 122 129
comptable 23 27 30 39 50 **264**-266 268 269
concurrence 46 62 63 73 103 113-116 130 138 149 194 249
confidentialité 54 64 72-74
configuration 20 98 140 141 156 163 167 175 179 209 211 213 216 235 247 248 257 258 262
connexion 25 **31** 33 37 61 106 107 111 148 161 222 223 227 230 232 262 271
console
consommateur 24
constructeur 19 20 107 127 131 137 147 157 224 227-230 230 268 270
contention (voir *concurrence*)
contexte 77 101 159 161 169 176 183 204 209 217 242 249
contiguïté 35 65-69 74 121 122 124 164 178 188 189 191 201 205 216
conversationnel 28 146 (voir *mode dialogué*)
coopération 56
Cortex 71 141 158 160
couche **54** 55 89 102 111 125 147 **226** 227 230
coût 19-21 30 222 263 266
CP/M 18 35 56 57 67 69 70 113 130 135 136 139 140 254
critique 80 234 269
cryptage 233 247
débit 23 50 80 124 155 181 235 265
débordement 124 206
décentralisé 45 68 70 71 104 105
DECnet 223 **225** 226
déconnexion 32
démarrage 34 72 163 197 211 212 253 **257**-262
démonté (voir *montage*)
déperdition 49 63 85 151 153 180 182 198 **264**
désallocation 43 **50** 51 53 248
descripteur 75 161 202-205 207 217-219 236-245 250 251
développement **26** 29 31 35 57 153 177 272
dialogue 56 89 101 110 130 140 154 158 **220**-227 229 230 252 253 258 259 (voir *asynchrone*, *synchrone*)
dictionnaire 147 (voir *répertoire*)
différé 31 97 103 266 269 (voir *mode*)
direct 116 **123** 127
disponibilité 23 35 45 55 72 100 147 213
disque 24 25 27 29-31 34 42 46-52 **64**-**67** 68 70-74 76 80 81 85 87 95-98 107-109 113-116 119-121 124 128 129 132 137 138 141 145 146 152 165 166 168-170 180 184 188 194 189 196 198 200 201 206 212 227 232 247 248 253 255 257-261 266 268
disquette 47 57 67-70 97 98 105 107 108 110 120 206 247 253 257 258
distant **32** 44 142 149 150 220-224 227-229 233
distribué 56 126 141 144 147-**149** **156** 158 159 167 208 222 227 229
distributeur 35 38 46 48 51 63 **75**-81 100 132 144 147 158 159 188 213 249 264
domaine 219 **244** 245
données (gestion des) 98 **102** **118** 119 125 143 249
DOS 28 35-37 56 74 75 92 131 137 139 140 144 153-155 164 168 179 193 194 201 209 217 224 250 252 266 (voir *VSE*)
drapeau 24 131
droit (voir *autorisation*)
DSA 56 **225** 227 230
dynamique 31 40 46 **48** 49 50 61 63 **65** 68 69 72-74 76-80 90 98 119 124 127 138 146 151 153 159 161 170 171 183 197 198 206 245 261 262 267
écran (voir *terminal interactif*)
édition de liens 16 42 61 90 131 157 164 193 243 267 268 271
emballement **181** 187 190 206
émulation 34 223 226 228 230
enchaînement 53 146 234
en ligne 104 107-109
enregistrement (voir *article*)
entrée des requêtes 37-40 43-46 52 61 82 91 138 150 159 224 227 271
entrée/sortie 17 42 47 50 63 65 75 77 78 80-83-85-87 93 96 99-101 104 108 109 130 134 137 147 161 168 174 184 194 195 234 247 248 265 266 268
entrée/sortie asynchrone 87
entrée/sortie lente **96** **137**
entrée/sortie logique 83 84 87 **89** **90**-92 94 119 147
entrée/sortie physique 73 80 82-**84** 86 87 89-92 94-96 99 102 119 159 184 189 264 265
entrée/sortie spoulée 83 84
entrée/sortie synchrone 75 87 88
entrée/sortie virtuelle 83 84 **98** 99 151 152
entrelacement 95
équilibrage 45 46 79 85 98 109 156 159
ergonomie 31
erreur 19 23 39 44 50 73 85 101 109 125 145 152 160 164 179 209 231 232 258 261 264 265
esclave 55 80 83 167 238 245 264
espace d'adresses (voir (*AS*)

étape **29** 30-32 **46**-53 61-65 73 74 79 109 131 138 139 142 165 201 221 251 254 255 258 264 265 267 270
état 43 55 63 64 75 99 100 103 108 109 143 156 160 167 196 198 235 239 249 257 261 262 267 271 (voir *mot d'état*)
étiquette 66 67 71 104 **108** **109** 247 249
étude 26
événement 63 75 79 81 89 101 130 131 266 269
exception (voir *interruption*)
exclusion mutuelle 131
exécutive 27 53 144 146 234
exécution **37**-44 46 48-51 53 55 61 64 65 68 73 75 76 78 86 89 90 99 101 109 111 112 134 135 138 143 146 155 159 160 163-165 176-178 181-183 193 201 203 210 214 215 258 262 265 267 270
exit 262
exploitation **25** 26 263
extension **65** **66**-70 74 95 117 200 202 215 234 248
facturation 19 26 39 109
faute de page 155 **180**-187 190 197
fenêtre 139 242
fermeture fichier 50 65 94 221
fichier 17 24 27 29 33 35 38-40 44 46-48 50-52 54 55 57 58 61 63-73 81 85 88-92 94-99 **102** **103**-117 119-129 131 132 135 140 143-145 147 159-161 188 189 194 196 197 202 203 207 208 210 211 214 215 221 223 224 227 228 230 232 233 239 244 247 248 250 254 255 258 260 261 264-266 268-272
fichier de commande 44 111 255 260
FIFO 40 86 132 159 185-187
file d'attente 74 86 100 145 146 158 159 161 166 250
filiation 64
FLEX 35 62 136 140
fonction 18 20 **23** 27 54 81 103 116 134 135 140 157 158 270 160 182 220 223 226 229
formattage 69
fournisseur 20 90 91 124
fragmentation 151 **165** **166** 170 178 179 205
frontal 46 74 97 98 170 222 224 225 227 229 230
gamme 36
GCOS3 28-31 35-42 44 45 47 48 50 51 53 56 58 60 63 66-68 71-74 76 78 80 82 86 87 89 95 98 103 104 107 113 116 122 123 127 128 131 132 136 137 139-142 144 147 151 155 165-170 216 218 225 228 235 250 252 255 260 264 267
GCOS6 35 37 92 97 110 140 148 249
GCOS7 30 35 36 38 38 41 50 55-57 78 82 86 130 132 139 140 190 250 251 255 258
GCOS8 18 36 140 155 184 190 201 203 205 206 218 219 245
génération 23 25 26 34 67 72 103 112 116 143 145 147 159 160 163 233 257 **260**-262 267
gérant d'appareil 74 85 93 110 135 136 222 229
gestion 16 **23** **25** 26 **27** 29 36 62 143 223 270
GIN 37 53 142 166
GOT 38 53
Guardian 52 56 58 67 108 110 141 147 184 225
hétérogène 224 229
hiérarchie 47 105 116 120 **126** 151 222 223 225 245 249
horloge 81 89 101 145
hors ligne 107
hyperpage **191** (voir *page*)
hyperviseur 55 56 **152**-155
icône 254 255
impression 42 50-53 97 112 134 140 271
imprimante 24 25 32 42 47-49 52 80 96-98 102 134 137 138 152 220 226 227 260 266
IMS 18 133 142 144 **145** **146** 149 210 261 270 272 224 226 237
incident (voir *erreur*)
incrémental 27
indexé 95 108 116 **120** **121**-125 127 128
industriel 230
initialisation 53 84 158 211 212 214 **257**-259
installation 27 79 113 117 137 148 163 183 191 194 257 **259**-263
instruction 17 22 64 77 81 83 85-87
intégrité 125 147
interactif (voir *mode dialogué*)
interblocage **49** 249
interface 55 57 135 142 143 160 161 204 220 226 229
interface externe 17 110
interface programmatique 17 81 103 110 118 130
interpréteur 135 138 244 258
interruption 17 34 35 37 54 57 63 74-76 80-82 85-89 94 99 **100** **101** 102 108 130 131 135-137 153 158-160 180 204 206 209 213 216-218 222 233 234 237 240-242 244 246 257 258 267
iRMX86 29 35 45 62 75 89 108 110 112 124 131 132 140 161 162 238
ISO
isoler 113 157 191 195 247
JCL 17 25 26 29 30 39 44 51 52 95 102 130 138 142 214 228 252-255 261
JES 37-43 45-47 53 97 99 109 133 138 141 142 212 224 227 237 259 261
jeton 161
journal 103 112 145 160 231 232
langage de commande (voir *JCL*)
langage de description 126
langage d'implémentation 22 36
langage de manipulation 126
langage de programmation 83 87 119 160 255
lecture **37**-41 43 51-53 134 135 138 142
légalité 20
librairie (voir *bibliothèque*)
licence 19
ligne/liaison 24 46 48 52 97 98 142 144 150 220-222 226 227 229
liste d'accès 73 **111** 112 **244** 246 247
local **32** 33 56 115 140 141 142 150 157 190-192 208 214 217 222 226-228 230 246 249
location 19 26
logiciel applicatif 16 17 26
logiciel de base 16 17 23
logimètre 27 **266** 267 269
logique 24 27 38 67 89 99 108 110 116 121 152 157 158 161 165 167 172 173 175 183 184 195 196 214 219 221-225 227 232 247 250 254
lot 28 **30**-32 61 78 79 (voir *mode différé*)

LRU 185 **186** **187** 195
machine virtuelle 79 113 151 **152**-156 198 217 236
macro 72 (voir *primitive*)
maintenance 18 19 22 **25**-27 54 99 152 158
maître 55 80 83 106 115 116 120 131 156 223 225 233-235 238 245 258 259 264
marché 20 22 118 140
matériel 15 17 19-21 23 50 55-57 61 62 73 75 81 84 101 102 119 130-134 152 154 156-161 167 169-171 173 178 181 194 201 204 205 216 220 222 231 233 237 239 250 251 260 262 269 272
membre 124 126
mémoire associative 151 173 174 177 178
mémoire centrale/physique/principale/réelle 2934 42 47-50 54 55 64 66-68 **74**-77 80 81 83 84 86 87 89 90 91 96 98 101 108 121 129 132 134-137 145-148 151 152 154 156 160 161 **163**-**165** 167 173 175-181 183 185-188 190-198 200 201 202 204 **205**-207 209-212 214-218 234-237 240 248-251 256 257 260 262 264-266
mémoire commune 157-160 191 201 210 211 238
mémoire locale 157 158
mémoire secondaire/auxiliaire/périphérique 6483 86 89-91 96 128 129 136 165 181 184 187 190 195 211 247 250
mémoire virtuelle 33 43 83 86 87 98 132 138 146 **151**-153 163 164 168 170 178 **179** **180** 182 184 185 191 193 196 198 **200**-**202** 203 208 209-211 214-218 236 250 251
menu 31 254 271
message 96 132 143 145-148 151 156 **157** 158 160 215 220 221 224 227 230 252 253 258 262 269
mesures 27 96 109 192 263 264 266-270 272
méthode d'accès 69 83 84 **90** 91 93 95 99 102 103 113 114 116 **118** 120-122 124 125 128 168 195 208 210 215 221 222 225 244 268
microprocesseur 36 158 172 239 241 246
microprogramme (microcode) 55 57 153 154 160 167 177 180 181 185
mise à jour 18 105 113 149 153 177 215 228 259 261 262 270
MMU **168** 172 173 203 204
mode d'accès **127** 150 247
mode dialogué **28** 29 **31**-33 **34**-37 40 **43**-45 48 61 62 65 76 77-80 84 142 150-153 155 167 170 189 192 224 232 242 254
mode différé **28** **29**-33 **34** 35 37 43 44 46 61 65 76 133 **142** 143 154 155 220 224 227 228 252 255 270
mode de fonctionnement 83 165 167 172-174 206 224 233 234 238 244 246 254 270 (voir *esclave*, *maître*, *privilégié*)
mode d'organisation **119**-122 **126** 127
module 22 73 137 146 167 168 182 184 203 207 211-213 260-262
modulaire 156 157
moniteur 53 135 143 144 227 232 233
montage 25 107-109
mot de passe 40 109 111 113 115 232 233 247
mot d'état 61 62 79 101 156 172 202 234 237
MS-DOS 27 35 39 44 45 56-58 67-69 71-74 87 95 97 100 104-106 108-111 113-115 124 128 131 135-137 139 140 163 226 253-260
multicritère 125
Multics 27-29 37 41 44 45 47 56 63 64 72 74 79 97 105 108 111 126 141 152 155 179 185 190 202 203 208 233 234 242 243 247 248 250 253-255 258 260
multifichier 108 **109**
multimicro 35 36 141
multimini 141
multiposte 57 **140**
multiprocesseur 58 75 80 100 141 145 234-236 264 265
multiprogrammation 40 43 58 75 95 96 **138**-141 148 156 163 167 176 178 181 201 211 250 269
multiSE 55
multisystème **20** **45** 56 58 115 140 141 158 248
multitâche 34 63 **138**-142 144 148 156 159
multitraitement **140** 141 156 250
multi-utilisateur 29 34 115 **149** 140 258
multivolume 66 71 **109**
MVS 18 19 28-30 35 36 38 39 43 46 47 49 50 52 53 56 57 62 63 72 74 79 85 87 92 93 96 98-100 104 106 109 115-117 124 132 133 137-142 145 146 153 154 163 167 171 183 185 187-190 191 193-196 198 199 201 203 208 **210**-219 224 228 234 235 237 246 248-251 255 258-262 264 266 269 270 252-254
nom 24 30 103-107 110 111 115 117 123 147 157 159 160 161 208 223 225 232 233 253 254
norme (voir *standard*)
noyau **53** **54**-58 61 63 73 74 75 77 81-84 87 90 91 97 100 102 104 118 136 137 141 144 147 150 154 157-159 164 167 198 211-214 216 221 231 233 234 237 239 258 259
objet 24 110 111 112 117 143 161 202 203 207 208 224 231 233 236 244-246 248 249
OCL 25 28 80 130 228 252 261
opérateur 23 **25** 26 30 38 46 108 232 253 248 259 262 266 267
optimisation 23 26 27 85 89 121 122 124 146 150 155 182 184 189 263 267 268
option 19 141 144 153 154 165 183 212 213 267
ordinateur (gros) 21 29 31 33 36 66 84 85 98 120 125 139 222-224 226 258
ordinateur (micro) 29 36 47 57 66 115 136 140 141 148 226 257 260
ordinateur (mini) 35 66 72 84 118 120 122 140 142 169 176 179 183 220 222-225 227 258
ordinateur (moyen) 21 29 31 36 66 84 85 98 125 226 258
ordinateur (petit) 31 33
ordonnancement 43 85
organisation **25** (voir *mode*)
OSI
OS9 **132**
OS1100 58 80 87 97 141 142 151 165 168 183 252 253
ouverture fichier 92 94 96 115 119 221 222 248
page 43 85 90 98 99 101 153 171 172 174 176-185 187-192 194 196-199 202 203 208 210 211 213 215 216 217 234 235 238 239 242 250 267

pagination 67 76 77 85 99 109 151 154 [illegible] 171 175 181-183 187 189 191-195 197 199 200 203 205 206 209 211-214 216 218 237 242 244 247 250 267 269
pagination statique 175 **178** **179** 181
pagination dynamique 172 176 **178** 179 **180** **181** 183
panne 45 68 149 156 158 209 216 223 257 264
paquet 48 89 157 230
parallélisme (voir *simultané*)
paramètre 243 246 253 254 255 260 267
partage 17 35 45-47 52 54 64 73 76 96 115 132 133 137-139 141 148 150-152 160 161 163 172 176 192 203 213 221 231 235 **249**-251
partition 56 57 67 **74** 79 144-146 155 **163**-**165** 167-**170** 171 175 177 179 182 183 205 206 209 236
partitionné 124 127 271
patch 261 262
performance 22 54 65 68 90 121 131 137 138 144 153 154 162 169 181 182 189 191 198 199 212 215 **263** 265 268-270
périphérique 24 38 46-52 54 58 61 63 64 73 80 81 83 84 86 94 96-98 100 102 104 108 **109** 110 115 119 120 134 137 140 147 152 153 156-158 161 174 199 201 214 220 221 223 240 254 257 259 260 265 267
permanent 48 109 220
Pick 35 140 155
physique 24 27 38 52 67 71 89 98 99 100 107 108 110 135 140 144 145 152 157 158 160 161 166 171 209 219 220 222 227 229 231 232 247 250 259 260 264 266 267
pile 100 101 107 168 192 203 207 216 218 219 **246** 247
piste 27 66 67 69 71 85 103 114 124 129
planification 26 28 37 38 **40**-**42** 43 46 47 49 51-54 61 63 76 146 151 168 190 258 259
pointeur 92 104 110 124 127 160 161 181 185 194 197 209 210 211 246 247
pool 74 159 161 164 170 197 210 214 267
portabilité 19 54
poste de travail (voir *terminal*)
post-pagination 184
POWER 37
pré-emption 35 49 160 249
prémontage 108
prépagination 183 184
préparation 25 26
primitive 62 72 81 86-89 99 101 106 110 113 118 124 131 139 208 226 250 255 261
Primos 35 40 44 51 58 64 65 79 107 108 111 120 127 129 140 141 184 201 253 254 258 260 262
priorité 34 35 **40** **41** 42 47 49 63 72 74 **75**-81 84 86 100 132 159 164 166 167 193 **238**
privé 47 73 **108** 109 146 191 195 199 201 210-214
privilège 54 82 112 131 132 153 154 165 202 211 213 233-237 **238** 241-243 245-248
procédure 26 124 157 203 206 220 221 234 236 241 243 246 255 261
procédure catalguée 26 **39** 52 246 255
processeur central 29 38 46-49 54-57 61 62 64 **75** 79 80 82 85-87 94-97 100 101 132 134 138 140 141 144 151-153 156-160 163 172 173 177 178 180-182 213 217 246 249 257 260 262 264 265 268 269
processeur d'entrées/sorties 47 80 100 156 162 174
processus 24 35 44 49 52 54 **61**-64 67 68 73 75-82 84 86 89 96 110 111 113 114 130-133 138-141 144 147 148 152 153 155-161 163 165 167-170 172 174 176 181 182 187 188 190 192 194 196-198 201-204 208 209 216-219 221 224 232 233 236-238 240-242 244-247 249-250
production 29 29 65 263
progiciel 20 21
programme 17 26 29 33 42 46 50 52 58 61 62 81 83 87-91 95 96 99 101 110 111 118 120 124 125 131 135 135 139 140 143-146 149 151 153 157 161 164 167 173 181-183 193 201 203 207 209-211 213 227 229 232 247 255 264 265 267 268 272
programme canal 85 86 99 100 174 177 184 188 199 268
Prologue 122 125 140
propriétaire **111**-114 126 160 248
protection 54 55 103 103 **110** **111**-113 131 132 138 144 151 164 193 194 201-203 206 207 212-214 231-235 237-240 242-244 246-248 262
protocole **220** 221 224 226 229 230
p-System 35 66
public **108** 109
pupitrage 23 **25** 28 34 42 50 63 73 80 107 109 153 228 252 258 259 261 266
quantum **48** **65** 67-69 74 76
racine 81 105 106 110 169 170 209 222
reconfiguration 109 147 262
recouvrement 81 136-138 151 167-170
redémarrage 51 68 101 117 147 149 213
redirection 44
redondance 35 72 73 126 155 242
ré entrant 90 147 211-213
registre 81 82 85 101 [illegible] 168-178 181 183 191 192 209 210 212 216-219 200 201 203 205 207 209 234-236
règle de recherche 256
relatif 95 108 116 **120** 123 127
relationnel 116 118 120 122 124 **126** 147 155
réorganisation 66 123 124 268
réparti (voir *distribué*)
répertoire 69-73 79 103 **104** **105**-108 110-112 115 117 119 123 126 147 152 161 165 205 212 221 243 256
reprise (voir *redémarrage*)
requête 23-25 28-31 34 **37** 39-46 53 55 61 62 72-74 85 86 99 100 109 112 130 132 142 146 151 166 170 174 244 264
réseau 38 45 46 56 67 89 107 115 116 **126** 132 140 141 145 148 150 155-157 161 208 **220**-230
résident 34 50 53 54 68 90 109 135-137 167 169 183 191 197 211 212 256 258 259
ressource 17 22 23 **24** 25 29 30 35 38 39 41-43 45 **46**-51 55 56 61-64 78-81 92 94 100 103 104 113 130 138 147 148 151 152 156 158 163 195 213 223-227 229 231 233 236 247-249 253 254 258 259 263-269 272
ré-utilisable 147 249

RSX 35 77 164 166 170 171 254
RTE-A 29 35-37 41 44 67 77 93 96 106 108 130 132 165 183 250 253-255 258
salaire 21
sauvegarde 25 27 101 176 195 212 249
schéma 125
scientifique 29 36 75 142
secteur 27 48 66-71 73 90 95 99 104 129 179 225 257 258
sécurité 23 54 55 64 72 73 92 93 103 108 109 125 231 232 234 241 243 247
segment 75 98 122 136 144 151 155 161 166 169 170 173 183 191 **200** 201 **202**-209 217-219 242-244 248 250 251 236 238-241
sémaphore 24 131 132 133 160 161 249
séquentiel 95 108 116 117 **120** 122 124 127 128 (voir *indexé*)
sérialisation 80 159 249
service 15 16 20 22 23 25 30-32 42 43 45 81 89 103 **130** 131 141 195 **226 227** 229-231 246 248 257 260 263
session **32** 33 **43** 44 52 61 142 147 148 221-224 227 229 232 233
shell 57 58 254 256
simultané 50 52 55 57 63 64 80 94 96 111 113 115 130 138-140 146 148-150 176 192 231 234 237 249
SNA 38 56 155 223 **224** 225 227 228 230
sortie des requêtes 15 38 39 45 46 50 **52** 53 82 138
soumission (voir *entrée*)
source 18 61 261 267 268 271
sous-système 20 **37** 38 40 43 **52** 53 78 82 84 91 94 97 108 115 144 146-149 174 222 227 252 253 259 262
spoule 39 45 46 52 96 97 **137** 168 152 155 164 248 267
SSII 20 147 266 268
standard 19 44 104 108 110 118 119 126 128 132 150 227-230 260
statique 46 **48** 49 61 **64** 68 70 74 75 119 124 **161 163 169 183 249 261**
structure 16 **17** **37** 43 54 82 104 105 108 110 116 120-122 125 126 128 129 134 137 144 146 147 157 160 167 178 192 197 203 205 207 209 211 212 214 220 222 225-228 230 231 259
suivi 27 266 267 270
sujet 161 231 232 248 249
superviseur 53-55 77 81 82 87 91 93 137 139 144 152 153 167 180 181 202 212 234 237 258 259 264
support 19 25
SVC 53 54 81 87 100 101 131 144 154 155 168 202 211 213 244
synchrone 33 45 89 130 132 158 184 196 221 223
synchronisation 35 46 64 81 88 89 130 131 133 146
SYSIN/SYSOUT 38 39 40 44 45 52 96-98 138 143 239 267
système 25 26 106 108 112 137 156 164 167 168 172 173 176 177 192 201 212 214 216 246-248 259 264 266 230 233 239
système spécialisé 45 46
table des pages/segments 153 154 174 176-181 191-193 196 198 199 200 202-206 208 209-213 216 217 219 234 236-238 244 245
tâche 24 34 **62** 63 100 132 139-142 144 155 156 158 193 213 214 216 218 (voir *processus*)
tambour 47 120 184
tampon 74 **84** 86 87 90-96 98 114 133 159 182 208 216
tassement 66 166 178 206
TDS 139 141 144 147 148 169 170 228
téléchargement **222** 228
télécommunication 21
télé-informatique 142 150
télématique 148 150
téléphone 32 33 130 141 157 220
télétraitement **149 150**
télévidage 228
temporaire 48 72 98 109 145 164
temps de réponse **30 31** 32 34 39 79 150 184 189 213
temps de service 85 87
temps écoulé 92 94-96 99 114 182 183 265
temps E/S 95 188 265
temps partagé **33** 35 36 40 76 78 139 **142** 143 147 **148** 150 167 220 252 255 270 271
temps processeur 42 50 92 94 95 128 139 153 155 170 182 183 188 195 197 206 248 253 264 266
temps réel **28** 29 **33 34**-36 57 77 78 84 139 143 **150** 167 170 222
terminaison **37** 41 42 45 49-51 63 77
terminal interactif 15 24 25 31 32 34 39 40 44 45 77 78 84 96 97 102 110 143-146 150 220-224 226 230 232 258 272
terminal lourd **32** 34 38 97 98 220-222 230
terminal virtuel
test 18 26 27 103 153 155 261 263
tolérance de panne 147 158
tourniquet 72 76 77 86
traduction d'adresse **86** 87 99 170 **171** **172**-178 181 182 192 200 201 203 205 206
tranche 63 **76**-80 264
transaction 20 28 31 **33** 35 80 81 139 141-**143**-147 **148** 149 151 158 185 210 215 227 270-272
translation 87 136 **165 167** 178 179 183 203 205
transparent 151 171 173 180 200
trappe (voir *interruption*)
travail 17 24 25 28 **29**-32 37 39-53 55 61 62 65 73 75 78 80-82 98 109 115 131 134 135 137-139 141-144 161 163-166 168 182 209 214 219 221 224 227 232 237 254 257 261 267 269-271
Tripos 68 71 73 110
trou 66 165 166 169
TSO 28 38 40 62 78 142-144 153 167 183 189 211 214 215 228 233 252 255 259 261 270
TSS 40 78 139 142 143 228 252
tube 63 **132** 158
unité 24 47 51 71 72 80 93 98 107 119 152 156 161 172 178 204 205 224 247 249 253 259 262
universitaire 42 52

Unix 29 35 36 41 44 45 57-59 63 64 66 70 71 77 91 100 104 108-110 112 117 120 128 130-132 140 155 157 158 167 185 252 254 256
utilisateur **15** **17** 22-24 26-31 33 34 37 40 42 44 50 51 54 56 63-65 67 **71**-73 78 79 81-83 87-89 91-93 96 97 99 101 103 **106** **107**-110**111**-113 115-117 119 120 123 124 127 130 131 135-137 139 140 142-144 146 148-152 156 160 161 164 165 167-170 172 173 176 177 190 192 197 201 208 210-214 216 223 224 228 229 231-234 236-238 244 247 248 252 253 254 256 258 260-264 266-270 272
utilitaire 17 26 44 54 99 135 253 259 268
verrou 46 49 **113**-115 159 234 **235** 236 **249** 250
version **18** 85 86 103 113-**115** 124 140 152 168 188 189 191 235 256 258 261 270
vidage 43 47 50 64 74 76 96-98 104 148 151 165-168 172 180 181 183-**185**-190 194-198 210 211 213-215 235 244 262 265
virtuel 62 99 152 155 174 **200** 223 (voir *entrée/sortie*, *mémoire*, *machine*)
VM 19 46 57 79 113 141 **152**-155 185 198 199 217 223 224 227 248
VMS 29 35 36 44-46 51 57 58 62 64-67 73 76-79 87-89 93 95 99 103 105-108 111 112 114 127 130 132 140 141 167 184 185 187 188 190 **196** **197** 198 201 216 232 233 239 246-249 251-255 264 267 268
volume 24 25 46 66-72 95 99 102-**107**-117 152 211 247 248 250 256
VS 19 (voir *DOS*)
VSE 18 35 66 73 163 164 260 268
VTAM 132 146 155 212 215 221 222 **224** 225 228 229 237 250 259 261 262
WS 181 188 **190** 191 193 195-197 199 267
XA 18 57 100 144 152 154 193 194 203 208 216 235

MASSON, Editeur.
120, boulevard Saint-Germain
Paris Cedex 06
Dépôt légal : juillet 1987

IMPRIMERIE LOUIS-JEAN
av. d'Embrun. 05002-GAP
Dépôt légal 242 — juin 1987

X0046498 7

DATE DE RETOUR
Veuillez rapporter ce volume avant ou la dernière date ci-dessous indiquée.

21 NOV. 1990			
23 MAI 1991			
10 [illegible]			
22 DEC. 1992			
[illegible]			
01 FEV. 2007			

ARCHITECTURE DES ORDINATEURS

J.-A. MONTAGNON

Cette série d'ouvrages se propose de décrire **l'architecture des systèmes informatiques** tant matérielle que logicielle, qu'elle soit centralisée ou répartie.

Le domaine est très vaste. Les différents tomes dégagent les concepts utilisés par les ordinateurs et les illustrent de façon aussi complète que possible.

Tome 1 : Sous-système central.
Tome 2 : Systèmes d'exploitation et extensions du matériel.
Tome 3 : Entrées-Sorties.
Tome 4 : Périphériques.
Tome 5 : Développement de programme.

Cette série intéresse :

- *les enseignants et les étudiants,*
- *les professionnels de l'informatique voulant élargir ou approfondir leur domaines de compétence,*
- *les esprits curieux qui souhaitent comprendre les ordinateurs à l'aide d'une approche structurée et mieux les utiliser,*
- *tous ceux qui veulent suivre l'évolution des systèmes informatiques.*

Le tome 2 décrit **les systèmes d'exploitation,** qu'ils soient conçus pour des micros, des minis ou des gros ordinateurs, qu'ils visent le traitement par lots, le transactionnel, le temps partagé ou le temps réel. Ce tome complète aussi le tome 1 en décrivant le support matériel des mécanismes de pagination, de segmentation. Les notions abordées sont abondamment illustrées par les microprocesseurs d'Intel, 8086, 80X86, les systèmes d'IBM, de Bull, de DEC, de Tandem, etc., sous MS-DOS, Unix, GCOS, Multics, VMS, MVS, DOS, VM, etc.

P7-DBE-359

9 782225 811845

ISBN: 2-225-81184-9