Collection connaissance du non-savoir
Essai de Some-Bits 74

Connaissance des ordinateurs

Gérard Ineichen mai 15, 1999

Table des Matières

1.- Aperçu historique des langages

Pris sous l'aspect du théoricien, les langages peuvent se répertorier en trois catégories qui se recoupent partiellement d'ailleurs.

Voyons donc ces trois groupes :

- les langages "préhistoriques", tels que les assembleurs et les langages d'assemblage. On les appelle souvent langages de bas niveau ou non évolués;
- les langages "moyenâgeux" ou de la "renaissance", tels que certains langages d'assemblage, "symboliques" ou algorithmiques;
- les langages des temps modernes, à savoir certains langages algorithmiques, spéciaux ou d'application.

Le praticien, quant à lui, devra toujours faire un peu de tout, et le choix du langage utilisé dépendra de critères fort différents suivant les applications : cela va de la disponibilité de tel ou tel compilateur jusqu'à l'efficacité tant à l'écriture du programme que du code généré, mais ne devrait jamais dépendre d'affinités particulières du programmeur envers tel ou tel langage.

Dans les pages qui suivent, l'ordre dans lequel les langages sont cités n'a aucun rapport avec le niveau du langage. Ils sont cités dans l'ordre où ils sont apparus. Cette liste ne se veut en aucun cas exhaustive.

Premier langage algorithmique

En 1948, K. Zuse définit le premier langage algorithmique qui n'a, d'ailleurs, jamais été implanté sur une machine. En 1954, Lanning et Zierler définissent le premier langage algorithmique utilisable, et d'ailleurs utilisé.

Fortran

Le document préliminaire définissant le Fortran a été publié en 1954 par IBM. Le premier compilateur, développé par J. Backus et un groupe d'IBM, utilisable sur IBM 704 a vu le jour en 1957. Les objectifs à atteindre et les caractéristiques principales de ce langage étaient :

- disposer d'un langage de résolution de problèmes mathématiques,
- disposer d'un langage de haut niveau indépendant de la machine,
- être un langage de haut niveau qui soit compilé (le Fortran était d'ailleurs le premier !),
- concept de sous-programmes et possibilité de compilation séparée,

- création de bibliothèques de programmes, sous-programmes et fonctions,
- variables simples et structurées (notion de tableaux),
- contrôle de l'allocation mémoire (statique, il est vrai) par les ordres COMMON et EQUIVALENCE,
- instructions d'entrées-sorties efficaces (mais complexes).

Plusieurs autres normes du Fortran ont été définies par la suite, et on distingue actuellement principalement le Fortran IV ou V et le Fortran 77 qui permet une meilleure structuration et l'utilisation de vraies chaînes de caractères.

Algol

En 1958, à l'initiative du professeur H. Ruetishauser, a eu lieu la première réunion à l'Ecole Polytechnique Fédérale de Zurich pour définir l'Algol 58, qui deviendra en 1960 l'Algol 60.

Algol est l'acronyme de "ALGOrithmic Language". Les objectifs de l'équipe zurichoise étaient d'être aussi proche que possible de la notation mathématique, utilisable pour décrire des algorithmes et traduisible en programme d'ordinateur. Ses caractéristiques sont :

- premier langage à être décrit formellement à l'aide de la notation BNF (Backus-Naur Form),
- les variables sont nécessairement déclarées; elles peuvent être de type entier, réel ou booléen,
- les instructions peuvent être composées, ce qui est intéressant dans les cas où plusieurs instructions dépendent d'un test, par exemple,
- notion de blocs et variables locales,
- allocation dynamique de tableaux et gestion dynamique de la mémoire par allocation et compactage des zones mémoires, ce qui permet l'utilisation de variables dimensionnées de taille variable définie lors de l'exécution seulement,
- notion de procédures récursives.

Le développement de ce compilateur a été fait par des scientifiques européens et américains, dont principalement H. Ruetishauser, P. Naur, J. Backus et A. Perlis.

Cobol

En 1956 est décrit le langage Flow-Matic qui est considéré comme ancêtre du Cobol. Le premier rapport complet et le premier compilateur Cobol sont prêts en 1960.

Le Cobol (acronyme de COmmon Business Oriented Language) est un langage orienté gestion dont la norme a été définie par le DOD (Department Of Defense), ayant pour but d'avoir un seul langage pour tous les ordinateurs de l'armée américaine qui soit indépendant du matériel. Les fonctions du programme devaient être réparties à l'intérieur de quatre divisions, elles-même subdivisées en sections. De plus, la notion de verbes (mots réservés du langage) était nouvelle.

Ses principales caractéristiques sont :

- la séparation en quatre divisions, à savoir :
- IDENTIFICATION DIVISION : sert à définir le nom externe du programme ou sous-programme, ainsi que les détails peu importants pour le compilateur, mais utiles lorsqu'une chaîne de programmes est développée : nom de l'auteur, remarques importantes quant à la "sécurité" telles que problèmes de validations, etc. Cette division est souvent considérée comme commentaires par les compilateurs Cobol,
- ENVIRONMENT DIVISION : sert à définir le contexte dans lequel on travaille, principalement en ce qui concerne les fichiers et le processeur. C'est parfois ici que l'on définit le choix de l'option de déverminage (debugging),
- DATA DIVISION : sert à définir toutes les variables qui seront utilisées (structure, type et taille). Elle est divisée en sections, notamment celles définissant les fichiers, les variables de travail, les écrans ou sorties formatées et les paramètres si c'est un sous-programme. Certains compilateurs Cobol autorisent en plus une description de l'écran de travail, avec quelques options telles que masque, validation à la lecture, etc.
- PROCEDURE DIVISION : contient le programme à proprement parler, divisé en sections et paragraphes,
- les types de données sont définis selon l'usage que l'on en fera, à savoir : DISPLAY (affichage, communication avec l'extérieur, où on définit tout jusqu'à la forme de sortie avec suppression des zéros non significatifs, ajout du sigle de la monnaie utilisée, etc.), binaire (COMPUTATIONAL) pour les calculs internes et INDEX pour les manipulations d'index dans les tables,
- la facilité de définition de format de sortie pour des fichiers ou des terminaux, mise en page sans grandes restrictions, mais parfois longue à définir,
- l'utilisation de différents types de fichiers : séquentiels, directs et séquentiels-indexés. (A noter que les périphériques sont considérés comme des fichiers par le Cobol).

Toute publication définissant le langage Cobol ou manuel d'utilisation du langage devrait contenir un "acknowledgement" tiré de la version originale de la définition du langage.

L'équipe qui a développé le Cobol était un groupe de la marine américaine, dont faisaient partie G. M. Hopper, R. Berner et C. Phillips.

LISP

Lors de la première conférence sur l'intelligence artificielle au Dartmouth College, en 1956, ont été définis les principaux fondements du langage Lisp (LISt Processor). Il s'agit donc essentiellement d'un langage destiné à des traitements non numériques qui a la particularité d'être interprété et non compilé. Il est utilisé surtout en intelligence artificielle, dans des domaines tels que la traduction automatique par exemple.

Le premier Lisp utilisable était disponible dès 1960, et le système Lisp 1.5, système complet, disponible dès 1962.

APL

En 1962, K. E. Iverson publie le premier document concernant l'APL : "A Programming Language". Le premier système complet APL date de 1966.

L'APL est un langage nouveau, sans ancêtre connu. Les objectifs de ce langage étaient de définir une nouvelle notation mathématique pour l'enseignement secondaire, qui devait être également un langage de programmation, ceci afin de pouvoir passer facilement des mathématiques à la programmation. A noter que l'objectif de la notation mathématique pour l'enseignement secondaire n'a jamais été atteint !

Les principales caractéristiques de ce langage, développé par K. E. Iverson et A. D. Falkoff, sont :

- les instructions sont exécutées, donc lues, de droite à gauche. Donc, pour l'expression :

$$4\,x^3 + 9\,x^2 - 2\,x + 3$$

on aura :

soit : 3 + (¯2 x X) + (9 x X * 2) + (4 x X * 3)

soit : 3 + X x ¯2 x 9 + X x 4

soit : + / 3 ¯2 9 4 x X * 0 1 2 3

- il s'agit d'un langage interprété et non pas compilé, utilisé presque exclusivement de manière interactive,

- les objets manipulés (donc les variables) sont des tableaux (listes, matrices, listes de matrices, etc.). Ainsi, l'addition A + B est une opération valable sur n'importe quel tableau A et B.

L'APL est resté un certain temps utilisable uniquement sur IBM, et est actuellement disponible sur plusieurs machines. Il nécessite une certaine place mémoire pour pouvoir être utilisé de manière efficace et il est indispensable de disposer de terminaux spéciaux, vu le jeu de caractères spécifique qu'il utilise.

Basic

En 1963 apparaît au Dartmouth College le Basic (Beginner's All-purpose Symbolic Instruction Code), langage dérivé des langages SCALP (Darmouth College), DOPE et DARSIMCO.

Ses objectifs sont d'offrir un langage simple pour des cours d'initiation à l'informatique, d'utiliser un ordinateur en mode partagé, interactif, et de constituer un système complet, à savoir un langage de contrôle et un langage de programmation. Il a été développé par T. Kurz et J. Kemeny. Ses caractéristiques principales sont :

- langage simple, interprété et non pas compilé,
- le nombre d'instructions du langage est réduit à environ quatorze instructions et environ sept commandes de "contrôle",
- système complet, indépendant et fermé : l'utilisateur n'a rien d'autre à connaître que le Basic. (A noter que sur les systèmes d'une certaine taille, il faut souvent appeler l'interpréteur Basic, et l'utilisateur n'est dans ce système fermé qu'à partir de ce moment).

Les instructions et variables permettant de traiter des chaînes de caractères ne faisaient pas partie du Basic tel qu'il a été défini en 1963. Beaucoup de constructeurs, surtout de micro-ordinateurs, ont vu l'avantage du Basic et ont construit des "machines Basic". Un manque de coordination et les limites du noyau de base vite atteintes ont fait qu'ils ont quasiment tous augmenté les possibilités du langage avant qu'une norme ne soit publiée. Le Basic a donc actuellement de multiples dialectes, ce qui a pour conséquence qu'un changement de machine ne peut quasiment jamais se faire sans adaptation aux particularités du Basic utilisé.

NPL

En 1963 également, un groupe d'utilisateurs de matériel IBM, Share-IBM, décide des exigences d'un nouveau langage de programmation, le New Programming Language. Une année plus tard, un document complet définissant ce langage est publié.

PL/I

En 1965, NPL doit, pour des raisons de nom déposé, changer de nom et devient PL/1 (Programming Language One). Il a été développé par le groupe Share-IBM et IBM. Ses objectifs étaient :

- le développement d'un langage qui rassemble tous les avantages du Fortran, de l'Algol et du Cobol,
- la conception d'un langage utilisable tant pour l'écriture d'applications que pour le développement système,

Pendant longtemps, le PL/1 n'était disponible que sur IBM, mais depuis quelques années il est aussi disponible sur d'autres matériels. Il en existe un certain nombre de sous-ensembles; il est donc indispensable, avant le transfert d'une application, de connaître les restrictions des sous-ensembles utilisés.

Algol

Une nouvelle norme d'Algol a été définie en 1968. Elle définit en fait un langage passablement différent de l'Algol 60.

Pascal

En 1969, à l'Ecole Polytechnique Fédérale de Zurich, a été défini le langage Pascal par Niklaus Wirth et son groupe. Le langage n'a été utilisable qu'en 1973, et sa caractéristique principale est le fait qu'il est prévu pour une "machine P" qui exécute du P-code, jeu d'instructions de bas niveau particulièrement adapté au Pascal.

Une des techniques souvent utilisées consiste à compiler le programme Pascal pour obtenir du P-code qui est ensuite interprété par un programme qui simule les instructions du P-code. A noter que le compilateur Pascal est lui-même écrit en Pascal.

Plusieurs langages en sont dérivés, en particulier des extensions devant permettre à plusieurs processus de travailler en synchronisation.

C

Le langage C a été décrit par Brian W. Kernighan et Dennis M. Rit chie dans le manuel "The C Programming Language" qui fait quasiment office de norme.

Le langage C est destiné à des implanteurs de systèmes d'exploitation, de drivers, de compilateurs et se veut être un langage ayant des structure de haut niveau et pouvant générer du code très proche de la machine.

Le système d'exploitation Unix a été écrit en C, à quelques modules du noyau près.

Ada

En 1978, le DOD (Department Of Defense) américain, constatant à nouveau le pluralisme des langages, fournit un certain nombre d'exigences pour définir un langage qui est mis au concours. En 1980, le dossier vert (car les définitions du langage ne portaient pas de références à leurs auteurs) est choisi et les spécifications publiées.

Le nom Ada est choisi en l'honneur de la Comtesse Ada-Augusta Lovelace (1815-1852) qui était la fille de Lord Byron et surtout l'égérie de Charles Babbage, "l'inventeur" du premier ordinateur théorique.

En principe, ce langage est complet et universel. Il est donc apte à traiter des problèmes mathématiques, de gestion, et se prête à la programmation de systèmes en multiprogrammation grâce à sa technique de synchronisation des processus par la méthode des "rendez-vous", de compilation séparée et de bibliothèque de sous-programmes. Comme tout sous-ensemble d'Ada n'est pas Ada, il existe actuellement plusieurs compilateurs qui ressemblent à Ada, mais le premier compilateur complet semble ne pas encore exister (été 1983).
Un tel compilateur existe en été 1986 sur les machines Vax (Digital) et sur les hauts de gamme Buroughs et CDC. Au centre de recherche Xerox de Palo Alto, une nouvelle machine spécialement conçue pour un environnement Ada a été développée.

2.- Généralités sur les langages et techniques d'assemblage

Le langage

Le dictionnaire Larousse définit le langage comme suit :

Langage : n. m. (de langue). Faculté que les hommes ont de communiquer entre eux et d'exprimer leur pensée au moyen de signes vocaux, qui peuvent éventuellement être transcrits.

Chomsky, du MIT (Massachusset's Institute of Technology), donne une définition plus scientifique :

Un langage L(G) généré par une grammaire G est l'ensemble de toutes les phrases possibles, c'est-à-dire l'ensemble de toutes les formes syntaxiques constituées uniquement de symboles terminaux.

Enfin, dans un ancien Larousse, la définition était encore plus simple :

Langage : emploi de la parole pour exprimer des idées. Pour clarifier ces définitions, donnons d'abord la définition de quelques mots :

- vocabulaire : ensemble non vide de symboles,
- chaîne : suite finie de symboles,
- grammaire :exemple, en Pascal, "begin", "if", "end" ou en Cobol "DIVISION", "OPEN", etc.), d'un ensemble de symboles non terminaux (par exemple, en Pascal, "instruction", "variable," ou en Cobol le verbe "MOVE ... TO ..."), d'un élément initial (par exemple : le programme) et d'un ensemble de règles (par exemple : formation des symboles non terminaux à partir des symboles terminaux).

Tout langage informatique est décrit de manière stricte, selon une méthode ou une autre. Citons, à titre d'exemples, les diagrammes syntaxiques (utilisés pour la définition du langage Pascal entre autres) et la notation BNF utilisée pour la définition de l'Algol et pour la publication des normes du PL/1. Voyons donc maintenant quelles sont les parties d'un compilateur.

L'analyse lexicale

L'analyseur lexical a pour rôle de lire le programme source, caractère par caractère, d'en effectuer le listage, d'enlever les informations inutiles pour le compilateur (les commentaires, les blancs), de former des entités qui seront des symboles terminaux (mots réservés, nombres, etc.) et enfin de détecter les erreurs lexicales telles que caractère illégal, nombre trop grand ou chaîne de caractères non fermée. Il fournit des entités à l'analyseur syntaxique.

L'analyse syntaxique

L'analyseur syntaxique prend les entités fournies par l'analyseur lexical et construit sur cette base l'arbre syntaxique. Il peut procéder selon une technique "top-down", c'est-à-dire, partir de la racine et construire jusqu'aux feuilles, ou, au contraire, partir des feuilles pour remonter à la racine (méthode "bottom-up").

Lors de la construction de l'arbre, il est possible qu'aucune règle de la syntaxe n'en permette la construction : on a alors une erreur syntaxique.

La partie sémantique

Une fois l'analyse syntaxique faite, le compilateur va effectuer certaines vérifications d'ordre sémantique, où il va tester une certaine cohérence du programme. C'est à ce moment que, en Cobol par exemple, le compilateur constatera qu'un fichier a été ouvert en écriture, et qu'il n'est utilisé que lors d'instructions "READ" ou qu'il n'est pas fermé. En Pascal, c'est cette partie du compilateur qui détectera les erreurs de type dans les affectations, par exemple.

Il est bien clair que cette partie du compilateur ne va pas tester le programme en simulant une exécution; elle ne fait le test que par recoupement des ordres du langage utilisé, de la table des symboles et de l'arbre syntaxique.

La table des symboles

Pour chaque référence à un symbole ou un identificateur, il sera nécessaire d'aller consulter la table des symboles, soit pour connaître le type ou le genre, soit pour contrôler l'unicité ou la déclaration. Il sera donc très important de bien structurer la table et d'y avoir accès rapidement.

Plusieurs méthodes existent, citons-en quelque-unes :

- recherche séquentielle : méthode lente, mais très facile à programmer,
- recherche binaire : méthode relativement rapide, mais qui présente l'inconvénient de nécessiter le stockage des symboles de manière ordonnée,
- organisation en arbre : méthode facile à implémenter si l'on dispose des pointeurs et/ou de la récursivité lorsque l'on écrit la routine de recherche, et qui présente l'avantage d'être très rapide si l'arbre est équilibré,
- technique de "hash-coding", qui est très rapide pour autant que la fonction de "hash" soit adaptée au cas traité et que la taille de la table soit suffisante pour éviter un trop grand nombre de collisions.

Souvent, on utilise des méthodes dérivées de celles-ci, voire même des méthodes hybrides.

Le traitement des erreurs

Le traitement des erreurs pose souvent de gros problèmes à l'implémenteur d'un langage. Citons d'abord, pour donner une idée, quelques types d'erreurs classiques :

- les erreurs lexicales : caractères illégal dans le programme source, tel qu'un signe "$" en Pascal ou en Cobol alors qu'il ne fait pas partie d'une chaîne de caractères, ou, si les nombres sont considérés comme symboles terminaux fournis par l'analyseur lexical à l'analyseur sémantique, un nombre trop grand ou composé de deux parties décimales,
- les erreurs syntaxiques : impossibilité de construire l'arbre syntaxique, soit qu'il y ait trop de symboles terminaux (cas de l'analyse "bottom-up") soit qu'il manque des symboles terminaux (cas "top-down"). Cette erreur s'obtient facilement en oubliant, par exemple, une parenthèse dans une expression,
- les erreurs sémantiques, dont le cas type est l'affectation d'une valeur numérique à une variable booléenne,
- enfin, les erreurs que le compilateur peut éventuellement prévoir, mais pas de manière absolue, et qui apparaîtront lors de l'exécution, telles que l'utilisation d'une variable non initialisée, division par zéro, etc.

Il est évident que tous les types d'erreur n'ont pas été énumérés ici, et qu'il ne s'agit que de quelques cas classiques. Certains compilateurs vont plus loin que d'autres dans l'analyse. Un des critères d'évaluation d'un compilateur peut être la clarté de ses messages d'erreurs.

Récupération en cas d'erreur

Un compilateur peut être jugé utilisable ou inutilisable en fonction des possibilités de récupération d'erreur dont il dispose. Citons quelques cas classiques, certains franchement mauvais, d'autres nettement meilleurs :

- le compilateur génère du code erroné, et continue la compilation sans message d'erreur,
- le compilateur se met en boucle ou s'arrête, sans message à l'utilisateur,
- le compilateur met un message à l'utilisateur, puis s'arrête (cas où il faut n+1 compilations, n étant le nombre d'erreurs se trouvant dans le programme source),
- le compilateur signale l'erreur, ne génère pas de code pour l'instruction incriminée, et continue,
- le compilateur affiche l'erreur, et arrête la génération de code, mais continue l'analyse du programme source jusqu'à la fin,
- le compilateur affiche qu'il trouve une erreur, indique comment il la corrige, et continue de générer du code pour en faire un module objet utilisable.

Cette dernière possibilité semble attrayante, et pourtant elle ne l'est pas nécessairement. Citons, à titre d'exemple, un cas où un essai de récupération a lieu, mais où les tables sont remplies de manière douteuse.

```
...
go to étiquette;              "étiquette" non défini,
                              le compilateur met un type
...                           numérique;
...                           go to numérique : erreur
...
                              numérique comme label est
...                           illégale
...
```

C'est volontairement que nous avons choisi ce cas, éventuellement limite, pour montrer cette récupération. Il est évident que le compilateur aurait pu faire mieux. Une conclusion importante est à tirer : la récupération et correction d'erreurs du programmeur est très difficile et souvent pas souhaitable, car elle va rarement dans le sens où l'auteur du programme voulait aller.

La génération de code

Afin qu'un compilateur soit utile, il doit générer du code pour que le programme écrit soit utilisable sur une machine donnée. On appelle le code généré par un compilateur "code objet",

- qu'il soit directement exécutable par génération d'un programme en binaire exécutable,
- qu'il faille le passer à travers un éditeur de liens, pour transformer le binaire relogeable en binaire absolu ou exécutable,
- ou encore qu'il génère du langage d'assemblage.

Le cas le plus fréquent est la génération de code objet sous forme relogeable, ce qui permet l'utilisation de modules compilés séparément, éventuellement en langages différents, et l'emploi de librairies de sous-programmes.

Optimisation

Le code généré peut souvent être, partiellement du moins, optimisé. Nous ne citons ici que quelques cas simples, étant entendu que cette optimisation peut être plus ou moins importante suivant le processeur utilisé :

- ne pas charger une variable dans un registre quand on vient d'y ranger la valeur,
- optimiser l'utilisation de tous les registres, et éviter d'utiliser des variables temporaires lorsque cela est possible,

- réutiliser les variables temporaires le plus souvent possible, et éviter d'en générer de nouvelles pour chaque expression source rencontrée,
- utilisation au mieux des possibilités de la machine en fonction du hardware,
- etc.

Il est évident que cette phase d'optimisation nécessitera un certain temps machine lors de la compilation, temps qui sera gagné lors de l'exécution et qui affectera peut-être aussi la taille du binaire exécutable. Un choix est donc à faire, et l'optimisation sera utile pour les programmes utilisés en exploitation, alors qu'elle ne sera pas toujours souhaitable lors de la mise au point du programme.

Enfin, la phase d'optimisation, bien qu'affectant le code généré, ne doit en aucun cas modifier les résultats obtenus par le programme objet généré.

Compilateur et/ou interpréteur

Ces deux termes sont souvent utilisés lors de la définition d'un langage, et, bien que certains langages soient parfois à cheval entre ces deux manières de faire, il est important de les différencier.

Le compilateur

Le compilateur traduit, comme nous l'avons vu précédemment, un langage source en un langage objet. Afin de pouvoir travailler correctement, il devra, en général, disposer en entrée d'un programme complet afin de pouvoir l'analyser et le traduire.

Le code généré, (code objet), peut être relogeable, absolu, du langage d'assemblage ou du code pour une pseudo-machine hypothétique (cas, par exemple, du Pascal P).

L'interpréteur

Contrairement au compilateur, l'interpréteur ne traduit pas du langage source en langage objet, mais en simule l'exécution au moyen de routines propres. Un interpréteur ne donne pas le contrôle au programme utilisateur par un "GO TO" dans du code utilisateur. Une des conséquences logiques de cette manière de faire est, souvent, un ralentissement à l'exécution, alors que le temps de compilation est économisé.

Notons encore que, dans certains cas, par exemple celui du Pascal P, le code généré est ensuite interprété. Cette technique peut permettre de disposer du code rapidement et facilement interprétable et, par conséquent, d'améliorer les performances du langage tout en en facilitant l'implémentation.

Le cross-compilateur

Une autre manière de faire, utilisée souvent en micro-informatique, consiste à écrire du code pour un processeur qui ne dispose pas toujours des moyens nécessaires à une compilation, sur un autre ordinateur, et de le compiler sur cette machine tierce. On parle alors de cross-compilation, ou compilation croisée. Tout ou partie des phases de création du code exécutable peut être fait sur une machine autre; c'est pourquoi on parlera parfois de cross-édition de liens.

L'édition de liens

L'éditeur de liens est utilisé pour la génération de code absolu ou exécutable à partir du code relogeable fourni par un compilateur et/ou des librairies de sous-programmes relogeables.

Son travail va être de rendre les adresses relogeables (donc définies relativement à une base propre au module chargé) définitives (donc utilisables directement par le processeur cible où le programme sera exécuté). Il créera également les liens avec les autres modules compilés séparément en complétant les références et, dans le cas de librairies de sous-programmes, en ne chargeant que les modules référencés. L'éditeur de liens fournira donc, à partir du code relogeable et des librairies, du code exécutable ou absolu et une carte d'édition des liens (ou map) où il indiquera quelles sont les routines chargées et à quelles adresses. Ce map n'est en général utile que lors de la recherche d'erreurs à l'exécution, et, trop souvent ignoré par l'utilisateur.

Langage machine

Un ordinateur est une machine qui résout des problèmes en utilisant un certain nombre d'instructions qui lui sont propres, telles que, par exemple, faire une addition, tester si un nombre vaut zéro, déplacer une donnée d'un endroit à un autre en mémoire, etc. L'ensemble de ces instructions qui sont reconnues par un ordinateur est ce que l'on appelle le "jeu d'instructions" qui sera, bien entendu, propre à chaque machine. La vitesse d'exécution des programmes dépendra donc d'une part de celle des instructions et, d'autre part, de la complexité et de l'adéquation du jeu d'instructions par rapport au problème à traiter.

L'ensemble des instructions de base forment le langage machine. Les concepteurs de l'architecture d'un ordinateur (designers) ont la possibilité de choisir un certain jeu d'instructions simples, performantes et de faible coût. Les langages machines sont trop simplistes, trop éloignés d'un langage naturel et par là-même sont inutilisables. La solution qui s'impose est celle d'imaginer un autre langage inexécutable mais plus proche de notre façon de parler. Mais pour exécuter un tel programme, il faut :

- soit remplacer chaque instruction par une suite d'instructions de l'autre langage puis les exécuter : traduction ou compilation.
- soit écrire un programme qui lit le programme et exécute une suite équivalente à l'instruction lue. Cette technique diffère en ce sens qu'aucun autre programme n'est écrit. On parle dans ce cas d'un interpréteur.

On peut alors parler d'une machine virtuelle à laquelle on s'adresse dans un langage évolué conçu de toute pièce. Il serait même possible de construire une machine dont les circuits exécutent directement les instructions du langage COBOL, FORTRAN ou PL/1. On peut concevoir un langage qui sera traduit ou interprété dans un langage lui-même traduit etc. On dit alors que la machine possède N niveaux (levels). Seul le top-level nous intéresse.

Assembleur pour une machine hypothétique

Prenons un exemple tiré de "Conception et implantation de langages de programmation" de D. Thalmann et B. Levrat. Nous allons considérer un assembleur pour une machine hypothétique. Voici les caractéristiques de ce dernier :

- registres : on n'en considère que trois : un registre arithmétique destiné au stockage des valeurs numériques en vue d'un traitement, un registre d'index destiné au stockage des adresses pour les déplacements en mémoire et un compteur d'instructions contenant l'adresse de l'instruction à exécuter. Notons-les A, X et CI respectivement.
- le format des instructions : dans tout langage assembleur, il faut fixer la dimension des champs : champs d'étiquette, d'opération, d'adresse et de commentaire.
- adressage : les différentes formes d'adressage que nous allons considérer sont les suivantes :
- direct : l'instruction utilise l'opérande situé à l'adresse spécifiée dans le champ d'adresse.
- indirect : l'instruction utilise l'opérande situé à l'adresse contenue à l'adresse spécifiée dans le champ d'adresse.
- immédiat : l'instruction utilise directement comme opérande la valeur spécifiée dans le champ d'adresse.
- dans le cas d'adressage indirect et direct, l'adresse peut être :
- absolue : déplacement relatif à l'adresse 0.
- relative : déplacement relatif à l'instruction en cours.
- indexée : l'adresse est obtenue par la somme d'une adresse spécifiée et du contenu du registre X. L'adresse spécifiée peut être absolue ou relative.

Les différentes manières d'adresser sont définies comme suit :

```
opération              adresse                  adressage

OP                         m                    direct
OP,*                       m                    indirect
OP,i                       m                    immédiat
OP                         K                    absolu
OP                         .+-n                 relatif
OP                         K,x                  indexé(absolu)
OP                         .+-n,x               indexé(relatif)

K est une adresse symbolique ou un nombre entier ou une expression

n est un nombre entier

. représente le compteur d'instructions
Les instructions sont les suivantes :

RES                  n     réserve n places
LDA                  m     (m) -> A
STA                  m     (A) -> m
LDX                  m     (m) -> X
STX                  m     (X) -> m
ADD                  m     (A)+(m) -> A
SUB                  m     (A)-(m) -> A
MUL                  m     (A)*(m) -> A
DIV                  m     (A)/(m) -> A
INX                        (X)+1 -> X et
                           saute 1 instruction si (X) = 0
JMP                  m     m -> CI
JSR                  m     (CI)+1 -> m puis m+1 -> CI
TE                   m     saute 1 instruction si (A) = (m)
TNE                  m     saute 1 instruction si (A) <> (m)
TG                   m     saute 1 instruction si (A) > (m)
TNG                  m     saute 1 instruction si (A) <= (m)

(A) signifie contenu de A
(m) signifie contenu du mot à l'adresse m
```

Exemple :

Calcul de z(z+3) pour z = 0 à 20 par pas de 2 et stockage dans 11 positions successives :

```
            LDX,i    -11
            LDA      z
LOOP        ADD,i    3
            MUL      z
            STA      W+11,X
            LDA      z
            ADD,i    2
            STA      z
            INX
            JMP LOOP
            ...
z           0
W           RES      11
```

Langage d'assemblage pour les instructions de contrôle

Pour décrire la manière dont les instructions d'un langage évolué peuvent être traduites en langage d'assemblage, donnons quelques exemples :

```
Les instructions sélectives

(PL/1)

 if k1 < k2 then begin;
                    k1 = 0;
                    k2 = k2 + 3;
                 end;

         LDA           A1,K1    /* on considère plusieurs registres
                                   arithmétiques pour ces exemples
         TG            A1,K2
         JMP           E1
         LDA,i         A2,0
         STA           A2,K1
         LDA           A3,K2
         ADD,i         A3,3
         STA           A3,K2
 E1      ....

 Les instructions répétitives

 do while k>0;
    t=t*k;
    k=k-2;
 end;


 E1         LDA         A1,K
            TG,i        A1,0
            JMP         E2
            LDA         A2,T
            MUL         A2,K
            STA         A2,T
            LDA         A3,K
            SUB,i       A3,2
            STA         A3,K
            JMP         E1
 E2         ...


 Les structures de données

 Les vecteurs :

 a(4) = 10;


    LDA,i   10
    LDX,i   A
    STA     3,X
A   RES     ...

 Les tableaux et les matrices sont utilisés de la manière suivante :

 a(i1,i2,i3,i4,...,i10) peut être accédé en faisant :

 LDA   I1
 MUL   N2
 ADD   I2
 MUL   N3
 ADD   I3
 ...
 ADD   I9
 MUL   N10   /* Ni est la taille de l'intervalle pour chaque indice
 ADD   I10
 ADD   BASE  /* une constante pour chaque calcul d'indice

 A ce propos, voir le cours de structures de données.

 ou alors plus subtil :
```

```
   LDX,i   -10
   MUL     N1+11,X
   ADD     I1+11,X
   INX
   JMP     .-3
```

Les sous-programmes

Les différents types de passage de paramètres sont :

par référence :

```
     P      0
            LDX    P
            LDA,*  0,X
            ADD,i  10
            STA,*  0,X
            ...
            JMP    1,X
```

à l'appel :

```
            LDA,i  3
            STA    N
            JSR    P
            N
            ...
```

par valeur :

```
       P    0
            LDX     P
            LDA,*   0,X
            STA     TEMP
            ADD,i   10
            STA     TEMP
            ...
            JMP     1,X
```

à l'appel on a :

```
            LDA,i   3
            STA     N
            JSR     P
            N
```

par résultat :

```
         P    0
              LDX   P
              LDA,* 0,X
              STA   TEMP
              ADD,i 10
              STA   TEMP
              ...
              LDA   TEMP
              STA,* 0,X
              ...
              JMP   1,X
```

appel identique.

Nous ne traiterons pas ici de l'optimisation du code, de la gestion de la mémoire, du traitement des expressions arithmétiques, etc.

3.- Editeur et chargeur

Introduction

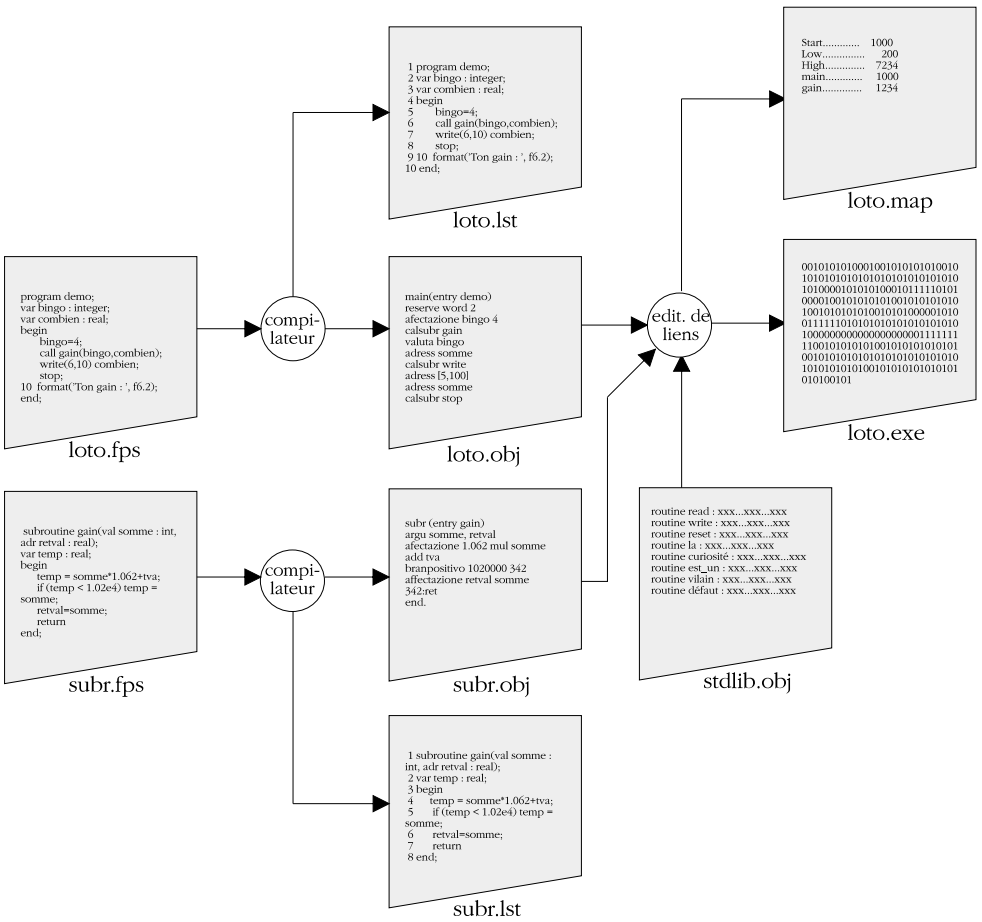
La plupart des programmes consistent en plus d'une procédure. Les compilateurs et les assembleurs généralement traduisent une procédure à la fois et rangent la sortie traduite dans un fichier.

Avant qu'un programme puisse être exécuté, toutes les procédures traduites doivent être recherchées et chaînées ensemble correctement.

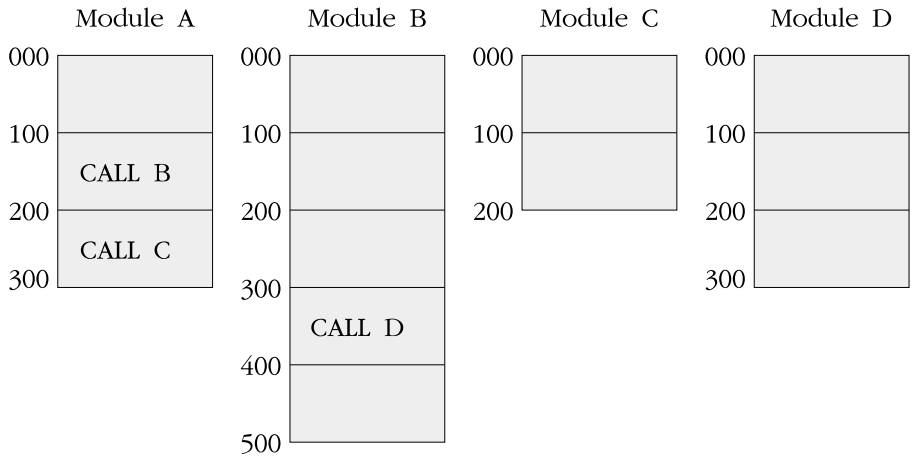
La traduction complète d'un programme source nécessite donc deux pas :

- compilation ou assemblage des procédures sources en créant du code objet
- édition des liens des modules objets.

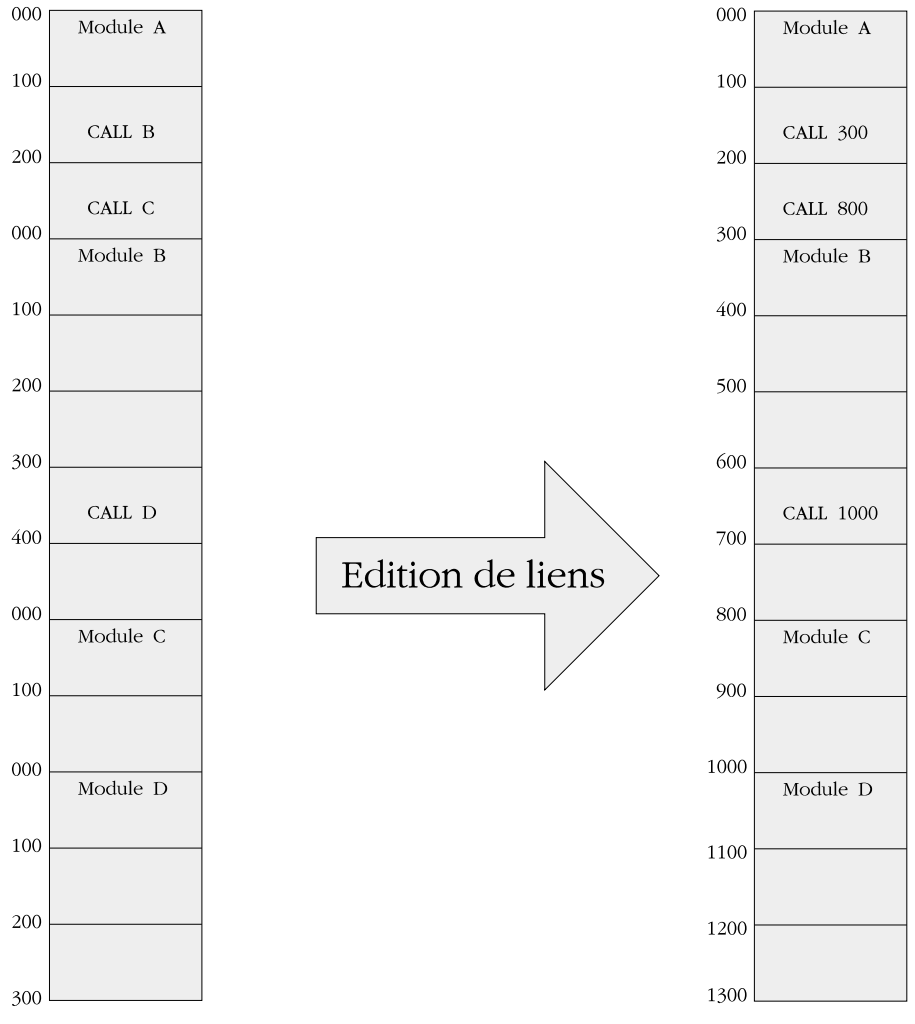
La fonction de l'éditeur de liens est de collecter les procédures traduites séparément et de les chaîner entre elles correctement pour être exécutées comme unité, appelée module absolu.



Considérons l'exemple suivant : un compilateur crée quatre modules objets différents; chacun débute à l'adresse 0.



Si on place les modules l'un derrière l'autre, on obtient la représentation à gauche. La partie droite représente les liens créés.



Pour exécuter le programme, il faut copier en mémoire le module absolu avant de commencer. Il y a de bonnes raisons pour que compilateurs et assembleurs traduisent chaque procédure source comme une entité séparée. Si un compilateur ou un assembleur lit une série de procédures sources et produit directement un programme en langage machine prêt à être exécuté, alors modifier une instruction dans une procédure source nécessite que toutes les procédures soient retraduites. Si l'édition des liens est utilisée, il est seulement nécessaire de retraduire la procédure modifiée puis de rechaîner tous les modules une nouvelle fois ce qui généralement est plus rapide.

Tâches effectuées par l'éditeur de liens

Au début de la première phase du processus d'assemblage, le compteur d'instructions est mis à zéro. Ce qui est équivalent à dire que le module objet sera placé à l'adresse zéro à reloger avant l'exécution. Les difficultés rencontrées seront alors :

- toutes les références à la mémoire échoueront car il y a différents espaces d'adresses. Exemple : JMP TO 200 dans un module et JMP TO 300 dans un autre module ne référencent pas le même espace d'adressage;
- sur une machine avec espace d'adresse segmenté, chaque module peut avoir son propre espace d'adresse en le plaçant dans son segment, ce qui simplifie la tâche de l'éditeur de liens;
- l'assembleur n'a pas le moyen de connaître les adresses des modules compilés séparément. Exemple : CALL B alors que B est inconnu !

La solution qui s'impose dans le cas d'un espace d'adresse linéaire est la suivante :

- construction d'une table de tous les modules objets et de leur longueur;
- à partir de cette table, affectation d'une adresse à chaque module objet;
- recherche de toutes les instructions qui contiennent une adresse mémoire et, à chacune, addition d'une constante de relocation égale à l'adresse de départ du module dans lequel elle se trouve;
- recherche de toutes les instructions qui font référence à d'autres procédures et insertion de l'adresse de celle-ci à la place de la référence.

Un registre de base pour l'adressage accélère l'édition de liens en réduisant le nombre d'instructions qui doivent être relogées. Comme la séquence d'appel charge un registre de base avec l'origine de la procédure appelée, celle-ci peut faire référence à ses instructions et données en spécifiant le déplacement depuis le début de la procédure indépendamment de l'endroit où la procédure est chargée.

Structure d'un module objet

Un module objet contient 6 parties :

- le nom du module et les informations nécessaires à l'éditeur de liens telles que la longueur des parties variables, la date de compilation, etc.,
- la liste des symboles définis dans le module que d'autres modules peuvent référencer, ainsi que des valeurs. Ex : si le module est une procédure appelée BIGBUG, alors la table des points d'entrée contiendra la chaîne de caractères "BIGBUG" suivie de l'adresse;
- la liste des symboles externes utilisés dans le module mais définis dans d'autres modules. L'éditeur de liens a besoin de cette liste pour insérer l'adresse correcte dans les instructions qui utilisent des symboles externes. Une procédure peut faire appel à d'autres procédures traduites indépendamment en les déclarant comme procédures externes;
- le code et les constantes. C'est la seule partie du module objet qui sera chargée en mémoire telle quelle. Comme l'éditeur de liens n'a aucun moyen de reconnaître parmi les données lesquelles appartiennent à une instruction ou une constante, un attribut est donné aux adresses à reloger;
- un dictionnaire de relocation par exemple, car il est possible qu'il faille référencer une variable ou procédure externe dont l'adresse n'est pas encore connue;
- une indication de la fin du module sous forme de checksum pour rechercher les erreurs lors de la lecture du module.

La plupart des éditeurs travaillent en deux passes :

- lire tous les modules objets et construire une table des noms des modules et leurs longueurs plus une table de symboles globale contenant toutes les "entries" et les références externes;
- les modules objets sont lus, relogés et chaînés en un module.

Relocation dynamique

Dans un environnement Time-Sharing, un programme peut être lu en mémoire, exécuté partiellement, réécrit sur disque, lu à nouveau et réexécuté. Dans le cas d'un système où plusieurs programmes sont dans ce cas, il est très difficile d'assurer qu'un programme soit relu dans les mêmes positions mémoire chaque fois. Toutes les adresses sont incorrectes mais surtout toutes les informations pour la relocation ont disparu.

Il y a en fait plusieurs solutions :

- quand l'éditeur de liens rajoute les espaces d'adresse séparés en un espace d'adresse linéaire, il crée en fait un nouvel espace d'adresse virtuel. Si l'espace d'adresse est segmenté, il est clair que les adresses correspondant aux noms symboliques sont déjà déterminées;
- utiliser un registre de relocation partagé par tous les programmes. Avant que le système d'exploitation ne passe le contrôle à un programme utilisateur, il initialise le registre de relocation à l'adresse mémoire physique du début du programme. Toutes les adresses sont ajoutées à celui-ci;
- toutes les adresses sont exprimées par rapport à l'adresse courante. Il suffira donc, lors de l'exécution, de l'additionner au Program Counter. Le programme pourra donc être chargé en mémoire sans être modifié.

Edition de liens dynamique

La méthode d'édition de liens ci-dessus a la particularité que les procédures qu'un programme peut appeler sont liées avant l'exécution. Sur un ordinateur avec mémoire virtuelle, une édition de liens complète avant le début de l'exécution ne prend pas tous les avantages de la mémoire virtuelle. Une méthode plus flexible pour lier les procédures compilées séparément est d'éditer les liens au moment du premier appel à la procédure : édition de liens dynamique. La position de chaque procédure compilée et rangée en mémoire est stockée quelque part de telle sorte que l'éditeur de liens puisse la retrouver quand cela est nécessaire. Associé à chaque programme, il y a un segment appelé segment d'édition des liens qui contient un bloc d'informations sur chaque procédure pouvant être appelée, contenant l'adresse et le nom de la procédure.

Quand l'édition de liens dynamique est utilisée, l'appel à la procédure est traduit en une instruction qui adresse indirectement le premier mot du bloc correspondant. Le traducteur remplit ce mot avec soit une adresse invalide, soit un bit qui force un arrêt (trap). Le système d'exploitation recherche dans le segment d'édition de liens le nom du module appelé et remplace l'adresse invalide par celle de la procédure. Les appels subséquents ne causeront plus cette erreur (appelée linkage fault) et l'appel sera fait sans intermédiaire. Ainsi l'exécution sera plus rapide (Cas de Multics).

Dans l'exemple ci-dessus, le premier appel à terre (call terre) aura pour effet de créer une erreur (* signifie uqe l'adresse est invalide). Le système d'exploitation reprend alors le contrôle, constate que le nom de la routine est là, et va chercher dans ses tables ou sur disque si la routine en question existe. Si oui, il la charge le cas échéannt, puis remplace le mot "terre" avec un pointeur illégal en une adresse réelle. Le second appel et les suivants iront alors tout de suite à la routine "terre".

4.- Notions de hardware

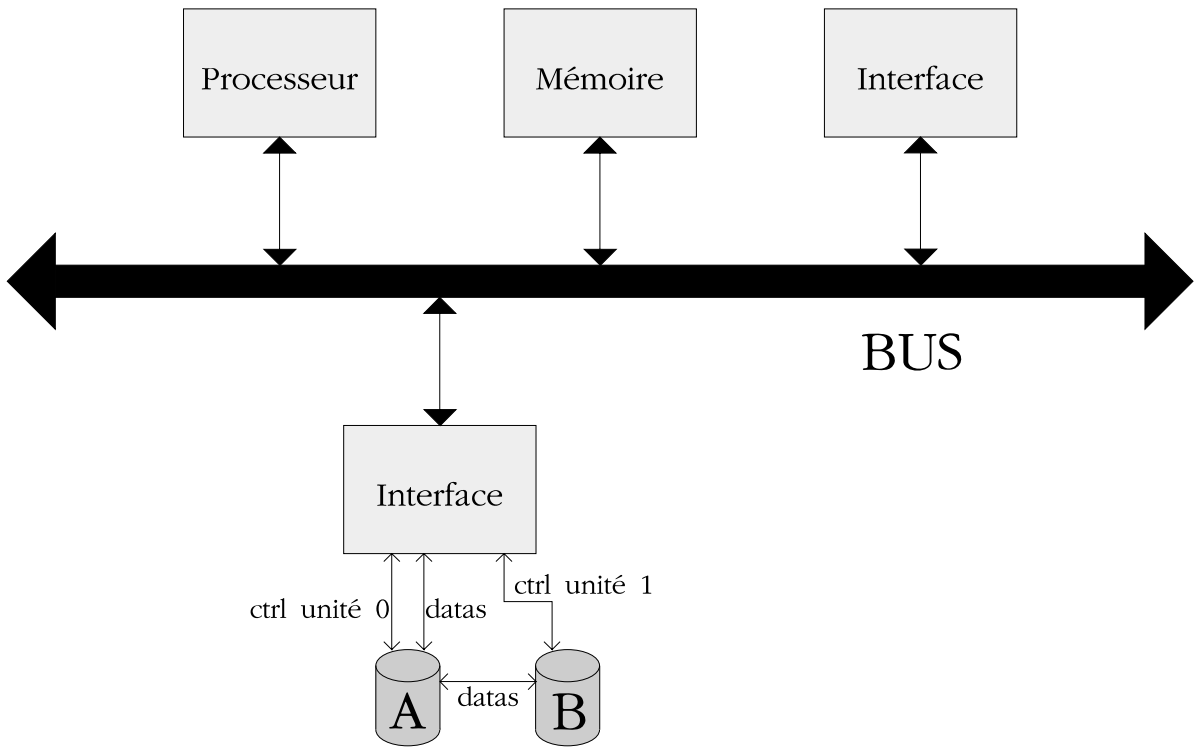
Nous n'allons pas détailler toutes les parties de hardware dans ce fascicule. Nous renvoyons le lecteur au chapitre sur les transmissions pour tout ce qui touche aux transmissions, au chapitre sur le parallélisme pour ce qui concerne les systèmes multiprocesseurs ou les processeurs frontaux, au chapitre sur la gestion de la mémoire pour le concept de mémoire virtuelle et pour les techniques d'adressage de la mémoire.

De plus, il existe une littérature importante, et, la technique évoluant à grande vitesse, ce fascicule ne saurait rester à jour indéfiniment.

Deux concepts pourtant vont être vus ici, de manière très succincte : la notion de daisy chain et la notion d'accès direct à la mémoire (DMA).

Daisy chain

Cette technique d'accès est surtout utilisée pour les disques et disquettes pour économiser des interfaces. Mais voyons plutôt par un schéma quelle en est l'idée :



Comme on peut le voir, une seule ligne de transmission de données va de l'interface sur la première unité (A), puis de là repart sur la seconde unité (B), voire sur une autre.

Un nombre limité de lignes va directement de l'interface à l'unité. On trouvera sur ce câble principalement le signal de sélection de l'unité, et, s'il s'agit d'une unité de disque, le contrôle du déplacement du bras.

Il sera ainsi possible d'utiliser une unité pendant que l'autre fait du déplacement de bras, mais il ne sera pas possible de lire ou d'écrire sur plus d'une unité à la fois. Cette technique est très souvent utilisée par les fabricants d'ordinateurs.

Accès direct à la mémoire

L'accès direct à la mémoire est utilisé par les interfaces ayant un certain débit pour éviter de trop pénaliser l'unité centrale avec les transferts d'entrées-sorties. Le processus qui a lieu est toujours sensiblement identique et les phases suivantes doivent avoir lieu :

- initialisation du canal d'accès direct à la mémoire, en donnant l'adresse du début du buffer en mémoire, la longueur de celui-ci et la direction de transfert,
- initialisation de l'interface pour démarrer le transfert,
- transférer, sans aide du CPU, chaque mot dès qu'un signal prêt est fourni par l'interface,
- après chaque transfert, incrémenter l'adresse du buffer, et tester si tout le buffer a été transféré,
- après le transfert du dernier mot, invalider le DMA et générer une interruption au CPU.

Comme on peut le voir, le CPU n'intervient que pour démarrer le transfert d'informations entre la mémoire et le périphérique, et ensuite il n'a plus qu'à attendre l'interruption de fin de transfert. Lorsque c'est le cas, il vérifie si le transfert a eu lieu sans erreur, et notifie le processus demandeur du transfert.

La gestion des adresses des buffers et des compteurs utilisés par le DMA peut être faite à plusieurs endroits : soit sur un contrôleur spécial du DMA, soit en mémoire centrale, soit encore sur l'interface lui-même. Il est dès lors difficile de donner une règle précise pour calculer le nombre de cycles du bus nécessaires pour un transfert DMA.

Un autre facteur qui peut influencer le ralentissement d'un système lors de l'activité du DMA est la manière dont il est implémenté :

- réception par le contrôleur DMA d'un signal indiquant que le périphérique est prêt pour un transfert DMA,
- demande à l'unité centrale de pouvoir disposer du bus (signal DMA request),
- attente de l'indication "bus libre",
- lecture, d'écriture et l'adresse mémoire,
- fin du transfert d'un mot par retour à l'état de repos de la demande de transfert DMA au CPU.

D'autres contrôleurs DMA utilisent des temps morts du bus pour transférer les informations de l'interface à la mémoire et il existe encore les techniques multibus qui sont souvent conçues en fonction de l'accès direct à la mémoire et des processeurs travaillant en parallèle. A noter que si les différentes opérations nécessaires au transfert d'un mot de l'interface à la mémoire n'ont pas pu avoir lieu entièrement avant que le mot suivant ne soit prêt, il y a un "overrun" dans le cas d'une lecture sur un périphérique, un "underrun" si c'est une écriture.

5.- Le système de gestion des fichiers

L'utilisation du disque, ou de la disquette, que nous considérerons comme équivalent quant à la structure logique, peut être plus ou moins compliquée. Certains utilitaires et parties de système d'exploitation peuvent rendre d'innombrables services en gardant à jour les informations quant à l'utilisation du disque.

Nous allons voir dans ce chapitre quelques manières de gérer ce stockage de masse, mais tout en restant assez général. Il est bien entendu que cette liste n'est pas exhaustive, ni détaillée au point que le lecteur reconnaîtra toutes les finesses de son système de gestion de fichiers.

Disque sans système particulier

Sur certains disques, lors des débuts de l'utilisation de ce type de périphériques, l'utilisateur devait maintenir lui-même, sur une feuille annexe, la table d'utilisation, en inscrivant les enregistrements ou les pistes occupés. Cette technique, pénible, mais utilisable lorsque l'on est utilisateur unique du support, devient trop lourde et risquée lors d'un partage de la place entre plusieurs personnes. C'est pourquoi on ne l'utilise plus que lors de transferts d'informations d'une machine à une autre sur disque souple. Dans ce cas, on précisera que le fichier commence à telle adresse et se termine à telle autre adresse.

Disque souple en format IBM 3741

Le format IBM 3741 est un format de disquette que l'on retrouve sur d'autres marques d'ordinateurs pour des raisons de compatibilité. Il définit un format logique sur un support physique donné, à savoir des disquettes huit pouces, simple face et simple densité.

Basé sur une répartition en 77 pistes de 26 secteurs chacune, il répartit l'information de manière très simple :

- la taille de chaque enregistrement physique est de 128 bytes,
- chaque enregistrement logique (ligne de code source dans le cas de fichiers "listables") occupe un enregistrement physique (facteur de blocage de un),.i.facteur de blocage

- l'organisation est séquentielle et contiguë, c'est à dire que la ligne suivante se trouve stockée dans l'enregistrement suivant,
- aucun compactage n'est prévu par l'utilisation d'un caractère de tabulation ou de répétition, par exemple,
- un fichier ne pourra être mis sur disque que si une place contiguë de cette taille au moins reste disponible,

- une réserve doit être allouée si l'on désire pouvoir modifier le fichier sans devoir le déplacer,
- une utilisation relativement intense du disque avec des fichiers de tailles variables engendrera une mauvaise utilisation de la place disque par création de multiples trous de petite taille, inutilisables de par leur petitesse,
- un catalogue des fichiers peut être écrit sur la piste zéro,
- deux pistes de "rechange" sont prévues parmi les 77 pour remplacer une ou deux éventuelles pistes défectueuses.

Afin de pouvoir réutiliser un disque écrit dans un format contigu sans cette perte de place par des trous de taille trop petite pour être utile, il est indispensable de procéder à un "ramassage des miettes" appelée "garbage collection". Cet utilitaire, parfois appelé "compress", va déplacer les fichiers de telle manière à créer un trou unique d'une taille aussi grande que possible; l'utilisation du disque pourra de nouveau être optimale pendant un certain temps, compte tenu de son contenu.

Il existe, bien sûr, d'autres manières plus efficaces de stocker de l'information sur disque. C'est ce que nous allons voir dans les paragraphes qui suivent.

Disquettes DOS sur PC compatibles

Sur la gamme des machines PC compatibles, on trouve principalement 4 formats de disquettes : les disquettes $5^{1/4}$ en haute et en basse densité, et les disquette $3^{1/2}$ en haute et basse densité.

Voici un tableau des caractéristiques de ces supports

Taille	capacité pistes	nombre de par piste	enregistrements
$5^{1/4}$ DD	360 Kb	40	9
$5^{1/4}$ HD	1,2 Mb	80	15
$3^{1/2}$ DD	720 Kb	80	9
$3^{1/2}$ HD	1,44 Mb	80	18

Si les disquettes de $3^{1/2}$ peuvent être lues indifféremment sur les lecteurs de 1.44 Mbytes, ce n'est pas le cas avec les $5^{1/4}$ qui doivent de préférence être utilisées que sur le bon lecteur. En aucun cas il ne faut formater des disquettes 360 sur un drive 1.2 ou le contraire, car le résultat est garanti avec problème à moyen terme.

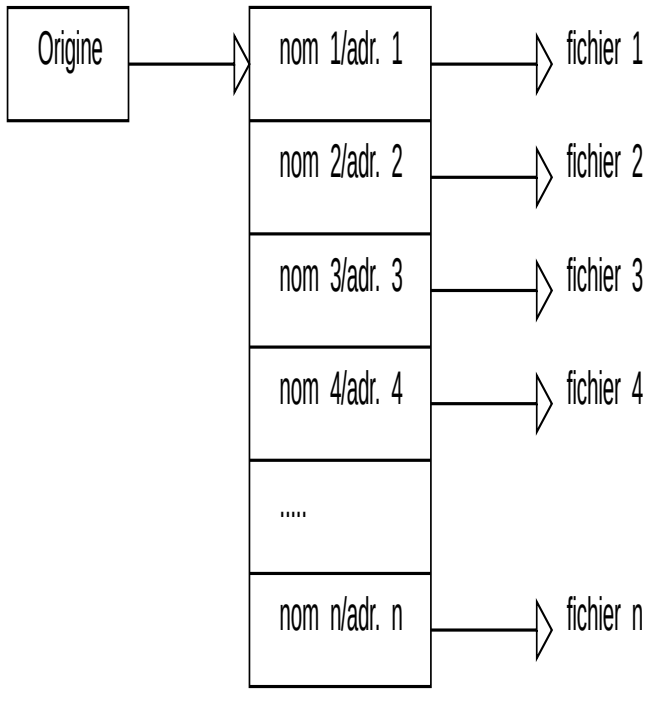
La structure des fichiers est, par contre, identique sur les quatre type de disquettes.

En outre, dès la version 5 du DOS, il existe la possibilité de "récupérer" une disquette formattée par erreur avec la commande "UNFORMAT". Cela signifie que la disquette n'est pas écrite entièrement, mais que seul un file system est installé avec un fichier contenant les caractéristique de l'ancienne structure. En principe, il est préférable de ne formater qu'à coup sûr, et avec la paramètre /U qui signifie que la disquette doit être reformatée entièrement, sans récupération. Ce n'est qu'à cette condition que certains cirus seront effacés à coup sûr de la disquette.

Gestion de fichiers en "open shop"

Dans un contexte en "open shop", c'est-à-dire où l'utilisateur a directement accès à la machine et à ses périphériques, on peut très bien considérer que chacun dispose de son propre disque, et que, par conséquent, aucune protection n'est nécessaire sous l'aspect logiciel, la confidentialité étant assurée par le retrait du support physique où se trouvent les informations.

On pourra donc stocker, à un endroit fixe sur disque, un catalogue où l'on indiquera le nom de chacun des fichiers, suivi de différentes informations, dont, en particulier, l'adresse de début. On aura donc une espèce d'annuaire, du même style que celui du téléphone qui aura un aspect de ce genre :

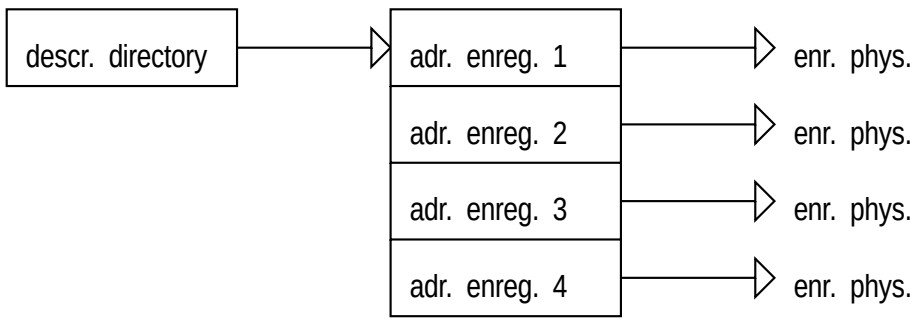


On associe donc à chaque nom de fichier une adresse disque où on pourra le trouver, ainsi que différents autres paramètres permettant de gérer correctement ces fichiers.

L'avantage principal est la possibilité pour l'utilisateur de donner un nom à son fichier selon ses propres règles sémantiques, en ne respectant que quelques règles de syntaxe relativement simples.

Fichiers "indexés"

L'organisation de fichiers de plus d'un enregistrement physique pose le problème de la manière de trouver les enregistrements suivants. Une des méthodes consiste à utiliser un enregistrement intermédiaire pour y stocker la liste des enregistrements physiques qui constituent le fichier. On aura donc quelque chose de ce genre :



Avec cette technique, les enregistrements physiques d'un fichier peuvent se trouver n'importe où sur le disque, et le temps d'accès à n'importe lequel d'entre eux est identique si l'on ne tient compte que du nombre d'accès au périphérique. Il est à noter que, pour accéder au premier enregistrement, il faudra lire deux enregistrements, catalogue non compris, et que, pour les fichiers courts (moins d'un enregistrement), les performances seront moindres et la place disque nécessaire doublée. Il y a donc un choix à faire, mais, pour cela, il faut que le système de gestion des fichiers permette d'autres organisations.

Fichiers indexés depuis le répertoire

On peut très bien organiser les fichiers de la même manière que décrit au paragraphe précédent, mais en allant stocker les adresses des enregistrements dans le directory lui-même. Cette méthode qui semble attrayante au premier abord présente toutefois un inconvénient majeur, à savoir augmenter démesurément la taille des répertoires. Elle ne sera que très rarement utilisée.

Fichiers séquentiels

Une autre méthode simple de gestion de fichiers consiste à utiliser les enregistrements de manière séquentielle et de ne pas garder de table des enregistrements, mais de les chaîner entre eux. Cette technique ne permettra pas de savoir rapidement où se trouve un enregistrement donné, mais permettra une meilleure gestion lors de l'utilisation de fichiers de programmes ou de données séquentiels. En voici un exemple schématisé :

Méthode hybride

Une méthode hybride entre celles énoncées ci-dessus permet une gestion qui se veut optimale : on stocke les adresses des "n" premiers enregistrements du fichier dans l'entrée du répertoire, et on n'indexe ou ne gère une structure séquentielle qu'à partir du n+1ème enregistrement. On a ainsi l'avantage d'avoir accès en un seul transfert disque (plus la lecture du directory, bien sûr) à une certaine quantité d'informations, dépendante de la taille de "n" d'une part et de celle de l'enregistrement d'autre part.

On gagne, par rapport à la méthode indexée, un enregistrement par fichier relativement court, alors qu'on ne perd que quelques mots dans le directory. Dans les autres cas, les caractéristiques sont sensiblement identiques à l'autre méthode choisie.

Fichiers à accès direct

Le fait de disposer d'un fichier à accès direct lors de la programmation signifie non pas que l'accès se fait directement au fichier, mais que le programmeur laisse le soin au système de gestion de fichier de lui simuler un accès direct. Souvent, il est nécessaire de disposer d'une structure indexée ou contiguë pour autoriser ce type d'accès; ce n'est toutefois pas toujours le cas.

Fichiers séquentiels-indexés

Les fichiers séquentiels-indexés sont des fichiers ou des groupes de fichiers pouvant avoir l'une ou l'autre des structures décrites ci-dessus, avec pour restriction la nécessité de pouvoir y accéder de manière directe.

Le logiciel permettra des recherches selon des clés non nécessairement numériques, et dont l'ordonnancement ne devra pas obligatoirement être séquentiel, en allant chercher dans un index la correspondance dans le fichier de données.

En général, on préférera organiser les tables d'index et les données dans un seul fichier ou en tout cas de telle manière à ce que l'utilisateur ne puisse pas voir qu'il y a en réalité plusieurs fichiers.

Les techniques de gestion de la table d'index sont nombreuses et ont quasiment toutes leurs partisans et leurs détracteurs. Citons juste quelques méthodes ici :

- par "hash-coding" de la chaîne de caractères formant l'index,
- par un système de listes,
- par un système de "hash-coding" avec des listes de débordement,
- par la technique des "B-trees",
- etc.

Ces techniques ne sont pas exclusives les unes des autres, et des mélanges sont possibles; un constructeur cite parfois comment il a implémenté les fichiers séquentiels-indexés, mais donne rarement l'algorithme exact utilisé.

Catalogues multiples

Dans un contexte time-sharing ou même dans un autre mode d'exploitation où plusieurs utilisateurs se partagent le même support physique, il peut être utile de disposer de plusieurs catalogues ou directoires.

Chacun de ces répertoires contiendra les fichiers d'un utilisateur ou d'un projet. Comme il est pénible de devoir donner chaque fois le nom complet du fichier (directory et fichier), il y aura moyen, sur tout bon système, de préciser un catalogue par défaut qui sera automatiquement concaténé avec les noms simples de fichiers. Il est bien sûr possible de disposer de catalogues dans des catalogues, chose que nous nommerons "sous-catalogues". Une bonne organisation permet de se promener d'un directory à un autre sans peine et doit permettre certaines options par défaut dans l'utilisation des noms de fichiers.

Catalogues et sous-catalogues

Une technique facilement utilisable dans un contexte multi-utilisateurs est la notion de catalogues et de sous-catalogues : on peut, par exemple, attribuer un catalogue par projet, chaque élément du projet étant dans un sous-catalogue avec tous les fichiers utiles.

Prenons, à titre d'exemple, un service informatique très simple qui s'occupe de trois applications distinctes : la facturation aux clients, les salaires et la gestion de l'horaire libre des employés. De plus, le programmeur travaille à la mise en place de deux nouvelles versions du logiciel utilisé. La structure disque choisie pourrait donc être :

fichier 2	fichier 2	fichier 2	fichier 2	fichier 2	fichier 2
fichier 3	fichier 3	fichier 3	fichier 3	fichier 3	fichier 3
...	...	...	...	...	...

Droits d'accès aux fichiers

Comme l'accès aux ressources doit pouvoir être restreint aux personnes autorisées, il faut trouver une astuce pour éviter des problèmes.

Plusieurs méthodes existent, les unes et les autres étant attrayantes sous certains aspects. Nous citerons ici deux grandes catégories de contrôle pour l'accès aux fichiers :

Contrôle par mot de passe

La méthode la plus simple à implémenter consiste à mettre un mot de passe sur un répertoire, et de l'exiger à chaque ouverture de fichier de ce répertoire. Cette manière de faire est parfois considérée comme obsolète.

On peut éventuellement différencier en deux ou trois catégories avec plusieurs mots de passe : l'un permettrait de tout faire sur les fichiers du directory, alors que l'autre ne permettrait que la lecture.

Listes de personnes autorisées

La technique des "access-list" permet une utilisation plus facile, avec possibilité de différenciation plus sélective : on cite, pour un objet donné du catalogue, les utilisateurs ayant des droits, et on les énumère.

Cette technique, avec possibilité de création de groupes de projets, par exemple, permet une saine gestion des accès à l'information en laissant une très large latitude au propriétaire de l'information dans le choix de ses "amis" (utilisateurs autorisés sélectivement).

Les systèmes modernes de gestion de fichiers tendent vers une méthode de ce type, avec possibilité de création de sous-catalogues à plusieurs niveaux.

Sécurité de l'information

La sécurité d'un file-system

Par sécurité de l'information en informatique, nous entendons deux points principalement; d'une part la conservation de l'information sans qu'il y ait de pertes dues à des chutes de tension ou des pannes, mêmes sporadiques, du système, par exemple, et, d'autre part, l'impossibilité pour des personnes non autorisées d'en prendre connaissance ou d'en modifier le contenu.

Sécurité du point de vue de la perte de l'information

Pour se préserver de pertes d'information dues soit à une malfonction du matériel, soit à des conditions extérieures (alimentation électrique, climatisation, ...), il faut copier toute l'information qui se trouve sur disque sur des bandes magnétiques à une périodicité donnée, une fois par semaine par exemple, avec une rotation des bobines sur plusieurs semaines, ainsi que chaque soir pour les fichiers modifiés depuis la dernière copie complète, avec une rotation des bobines sur une semaine.

Ces copies doivent être faites systématiquement pour tous les disques utilisés; il est préférable de pouvoir le faire pendant l'exploitation lorsque cela n'a qu'un aspect de backup. Elles permettent de ne perdre, au pire des cas, qu'une journée de travail.

Sécurité du point de vue de l'accès aux fichiers

Le principe de base utilisé dans le cadre de cette sécurité est l'utilisation de mots de passe. Ces mots de passe peuvent avoir un certain nombre de caractères, (six par exemple, ce qui devrait être un minimum), et doivent être tapés "en aveugle", c'est-à-dire qu'ils n'apparaissent pas sur l'écran. Chaque identification peut avoir plusieurs mots de passe, bien que, en règle générale, un seul soit attribué, le second étant "blanc". Il est bien clair que la personne ayant pu se procurer le premier de ces mots de passe aura accès aux fichiers au même titre que le propriétaire : il pourra modifier les fichiers, les clefs d'accès aux fichiers, etc.

Une autre méthode consiste à n'utiliser que des listes d'accès et à ne pas considérer les mots de passe. Le contrôle n'a donc lieu, dans ce cas, qu'au moment où l'utilisateur s'annonce au système, son mot de passe ou la méthode de validation du début de chaque session n'en devient que plus importante.

6.- Gestion de la mémoire

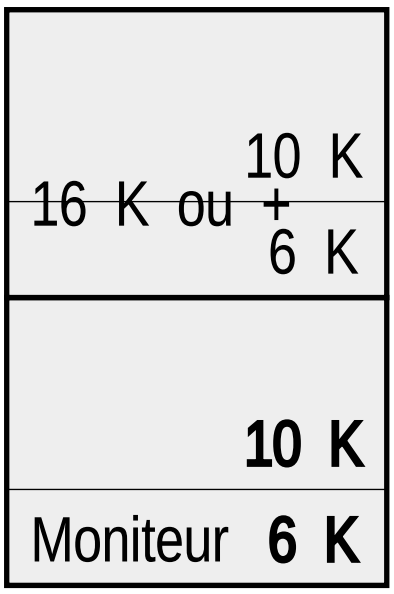
Dans la plupart des systèmes d'exploitation, qu'il s'agisse de micros, minis ou au contraire de grosses machines, un problème important apparaît : comment gérer la mémoire à disposition ? Les cas existants sont en général :

- le système d'adressage de la machine est à 16 bits, ce qui limite, théoriquement, les possibilités d'adressage à 64 K mots, et on aimerait disposer de plus de 64 K mots de mémoire,
- le système d'adressage de la machine est à plus de 16 bits et on désire utiliser des zones d'adressage où aucune mémoire physique n'existe,
- on désire travailler dans un contexte de multiprogrammation, chaque processus ayant sa propre zone mémoire répondant toujours à la même adresse,
- on travaille sur un système time-sharing, et chaque utilisateur doit pouvoir disposer du même jeu de programmes et utilitaires qui utilisent des zones mémoire fixes,
- on désire écrire des programmes nécessitant des quantités de mémoire plus grandes que ce que l'on a réellement à disposition, et ce sans devoir "bricoler" avec des overlays ou des fichiers.

Dans tous ces cas, un problème relativement complexe est à résoudre, à savoir la gestion de la mémoire. Pour que le programmeur puisse être plus efficace, il faut que cette gestion soit transparente à l'utilisateur, c'est-à-dire qu'il ne doit pas s'occuper de ce problème lors de son travail. Il est toutefois utile de connaître le mécanisme de fonctionnement de cette gestion, afin de choisir une méthode de programmation pouvant diminuer l'"overhead" du système de gestion de la mémoire. Dans les pages qui suivent, nous allons décrire quelques méthodes de gestion, certaines rudimentaires et d'autres sophistiquées. Nous allons traiter d'abord un exemple relativement archaïque, utilisé sur certains anciens mini-ordinateurs en time-sharing.

Premier exemple : par partition

Prenons, à titre d'exemple, une machine qui a au maximum 32 K mots de 16 bits de mémoire centrale, sans gestion spéciale de l'adressage. Nous désirons travailler dans un contexte time-sharing, avec des partitions utilisateur de 6 K, 10 K ou 16 K, tout en ayant, bien sûr, un moniteur qui soit résident. On peut donc répartir la mémoire en partitions comme suit :



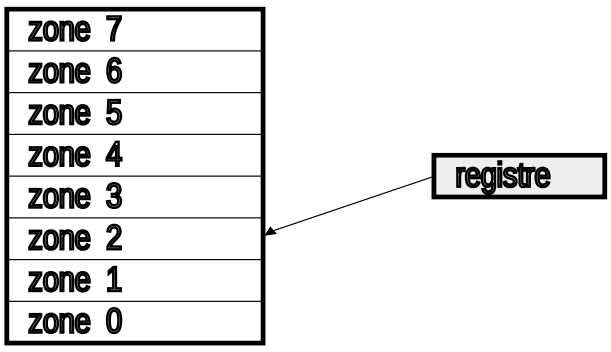
Cette technique, qui semble pourtant simple, a de multiples inconvénients dont les principaux sont :

- un programme qui a été chargé (édition de liens) pour être exécuté dans une partition donnée, devra toujours être rechargé dans cette même partition pour l'exécution, même lors de time-slices ultérieurs,
- si on n'a pas de moyen de savoir si une adresse correspond bien à une partition donnée, il faut charger toute la partition allouée avant de passer le contrôle au programme utilisateur,
- pour une partition donnée, les utilisateurs devront attendre chacun leur tour, même si une autre partition est libre à un moment donné,
- la répartition du temps machine peut devenir inéquitable si un seul utilisateur demande une partition, et si beaucoup en veulent une autre : le système essaiera, en effet, de charger en accès direct à la mémoire, une partition pendant que le contrôle est donné à un utilisateur,
- la taille d'un travail utilisateur est fixée lors de l'édition de liens à une des possibilités standard : 6, 10 ou 16 K dans notre exemple,
- la protection (confidentialité) entre utilisateurs ne peut pas être garantie entièrement, car le système n'interdit pas nécessairement la consultation des autres partitions; elle est toutefois aléatoire, car l'utilisateur ne peut consulter que la partition chargée à un moment donné, sans savoir où il se trouve dans son time-slice.

Un avantage certain toutefois : le système de protection entre utilisateurs est très simple, car un système d'interdiction de modification d'une zone mémoire non attribuée suffit pour cette méthode de gestion de la mémoire.

Exemple avec des bancs de mémoire

Dans cet exemple, on admettra que l'on dispose d'un espace d'adressage de 16 bits, à savoir 64 K mots, et une mémoire de 256 K. Le problème à résoudre est donc de pouvoir adresser plus de mémoire qu'il n'est théoriquement possible. Voyons donc une solution :



La technique d'adressage va être choisie comme suit : si le bit de poids fort de l'adresse est à zéro, on adressera la zone zéro, si le bit de poids fort est à un, on étendra le nombre de bits d'adresse à dix-huit, en prenant les bits 2^17, 2^16 et 2^15 dans le registre de zone, les bits de poids 2^14 à 2^0 étant pris de l'adresse même.

Dans un contexte de multiprogrammation, on peut très bien prendre le "bank" zéro (zone zéro) pour le moniteur, et mettre les autres programmes dans les "banks" 1 à 7.

Les avantages de cette méthode de gestion de la mémoire sont :

- jusqu'à sept zones (dans cet exemple) de mémoire totalement indépendantes les unes des autres, et chargées en mémoire physique, (la huitième étant ici la zone commune),
- chargement des autres partitions physiques en simultané pour autant que le hardware permette un adressage différencié pour l'accès direct à la mémoire,
- un programme peut être chargé dans n'importe laquelle des partitions, car elles répondent toutes à la même adresse.

Les inconvénients, pourtant, existent aussi :

- nécessité d'un moniteur situé dans le "bank" zéro qui se charge du changement de contexte, à savoir ici le chargement du registre de "bank",
- communication d'une zone à une autre impossible sans qu'il y ait passage par la zone zéro.

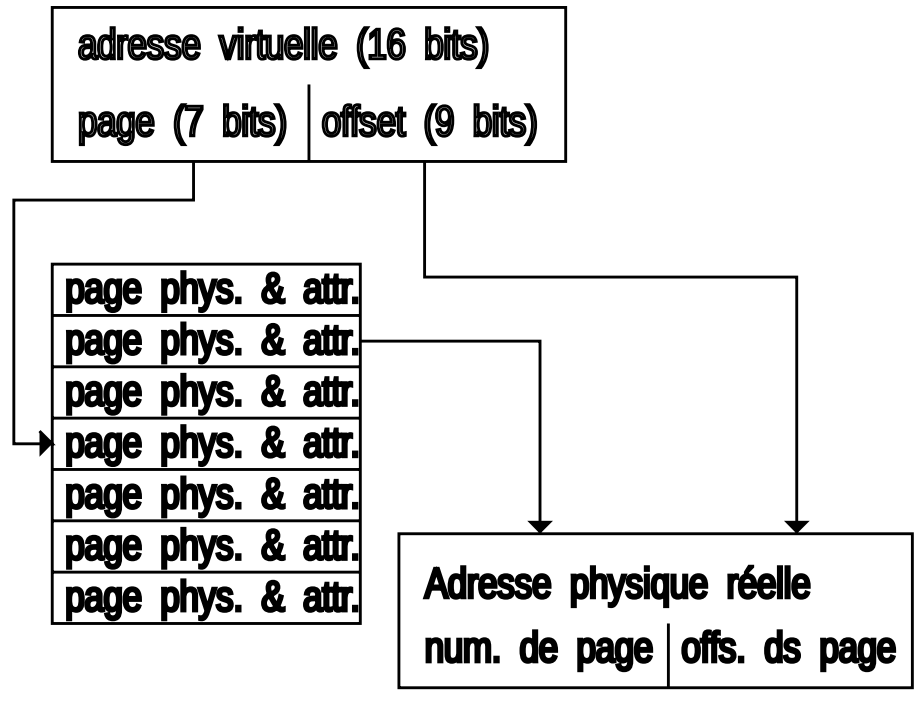
Notons enfin que cette manière de faire est très facile à implémenter sous l'aspect hardware, car il suffit d'un registre dans le décodage d'adresse qu'il faut pouvoir charger par programme en cours d'exécution.

Mémoire virtuelle

Cette technique permet de ne pas mettre en mémoire centrale tout le programme à exécuter, mais une partie seulement. La manière de choisir quelle partie est résidente est décrite plus loin dans l'algorithme de gestion de mémoire virtuelle. A noter que la notion de mémoire virtuelle ne signifie pas que la mémoire sera simulée, mais indique qu'il y aura une translation d'adresse entre celle connue par le programme et celle qui désigne une position mémoire. On peut rapprocher cette manière de faire de celle consistant à fragmenter le programme en différents overlays, avec toutefois une grande différence : c'est le programmeur qui décide quelle partie doit être chargée en mémoire lorsqu'il utilise des overlays, alors qu'il ignore ce qui est résident ou non dans un contexte de mémoire virtuelle.

Voyons donc deux exemples d'implémentation de mémoire virtuelle sur des mini-ordinateurs :

Le premier exemple est un cas d'ordinateur de taille de mot de 16 bits, avec une taille réellement adressable de 64 K mots de 16 bits. Le constructeur a décidé que la taille d'une page (zone unitaire de pagination) serait de 512 mots. Le décodage d'adresse aura donc lieu comme suit :



Les avantages d'une telle manière de gérer la mémoire sont multiples, nous n'en citerons qu'une partie ici et y reviendrons lorsque nous parlerons de la manière de choisir les pages résidente :

- la mémoire est attribuée de façon contiguë logiquement, mais de manière aléatoire réellement; on n'aura donc pas recours à des algorithmes de compactage de zones mémoires pour obtenir des "trous" de taille acceptable,

- le programme n'a pas besoin d'être chargé en entier pour pouvoir être exécuté, car il suffit d'utiliser le bit d'erreur d'adressage pour savoir si on doit modifier le contexte mémoire,
- les options hardware fournies en général lorsque ce type de mécanisme est implémenté, permettent de protéger efficacement les zones où le programme utilisateur ne doit pas avoir accès.

A remarquer que cet exemple traite le cas d'une machine avec espace d'adressage de 64 K et une taille physique de mémoire de 256 K.

Le second exemple montre le cas où la mémoire logique de l'utilisateur est segmentée par 64 K, avec des adresses à 28 bits et des tests de droits d'accès différenciés par segment en fonction du niveau auquel le processeur exécute l'instruction. Dans cet exemple, les pages mémoire ont une taille d'un K mots, l'adresse définitive ayant 22 bits. On constate donc que l'espace d'adressage à disposition de l'utilisateur est plus grand que la taille maximum de la mémoire physique.

Dans ces deux exemples, nous supposions en général que la page était en mémoire physique. Comme nous n'avons pas assez de mémoire physique, ce ne sera pas toujours le cas. Lorsqu'une référence à une position mémoire qui n'est pas résidente a lieu, un "page fault" est généré. Cette condition interrompt l'instruction en cours d'exécution et une routine spéciale teste quelle erreur a eu lieu. Quand elle constate qu'il s'agit d'une position mémoire non résidente référencée, elle charge la page depuis le stockage de masse, valide l'accès dans la table et rend le contrôle au programme interrompu qui devra réexécuter l'instruction qui a créé l'erreur.

La technique choisie ici est donc ce que nous appellerons "demand paging", car une page mémoire n'est chargée que lorsqu'elle est demandée par une référence.

Adressage

L'électronique supplémentaire pour pouvoir utiliser une technique de mémoire virtuelle devra comporter différentes possibilités dont la plus importante est une logique de conversion hardware des adresses au niveau du bus du système : l'unité centrale, en effet, ne doit pas remarquer que l'adresse réelle de l'information en mémoire est différente de celle qu'elle a utilisée. Au cas où la page référencée n'est pas résidente, l'électronique devra générer une erreur d'adressage qui doit être signalée au processeur afin qu'il entre dans une routine récupérant le cas en mettant à disposition la position mémoire demandée et réexécutant l'instruction avortée.

Particularités du module de gestion de mémoire

Afin de pouvoir gérer correctement l'espace de mémoire physique à disposition, certains tests supplémentaires sont nécessaires au niveau d'un module de gestion de mémoire virtuelle : il est, en effet, indispensable de savoir si une page mémoire a été référencée depuis le dernier point fixe et si elle a été modifiée. Ces informations permettront de savoir si la page doit être écrite sur le stockage de masse ou si la copie qui s'y trouve est toujours actuelle (on appelle cela parfois le problème des données rances, car les copies disque et mémoire sont différentes).

Hardware supplémentaire utile

Enfin, pour une saine gestion de la mémoire dans un contexte hostile (contexte de mise au point de programmes et d'essais de logiciels non certifiés), il est souvent utile de pouvoir créer une erreur lorsque le programme à tester dépasse certaines bornes par l'utilisation, par exemple, d'éléments de tableau avec un indice trop grand ou trop petit, d'adressage indirect non ou mal initialisé ou tout autre dépassement des bornes déclarées lors de l'édition de liens.

Pour cela, il faut rajouter une possibilité d'interdire l'écriture en mémoire hors de ces bornes. Si toutefois tel devait être le cas, une erreur doit être signalée, hardware d'abord, puis par le système d'exploitation qui en informera l'utilisateur au moyen d'un message suffisamment clair.

Contexte multi-utilisateurs

Enfin, afin de pouvoir satisfaire la confidentialité d'une application dans un contexte multi-utilisateurs, il faut interdire l'accès, même en lecture seulement, d'informations n'appartenant pas au programme en cours d'exécution. Pour cela, la plupart des modules de gestion de mémoire virtuelle disposent d'une possibilité de hiérarchisation de l'accès aux différentes pages en laissant des droits d'accès différenciés : lecture, écriture, exécution ou toute combinaison de ces droits.

Paging : algorithme "LRU"

Le gros problème qui apparaît très vite en utilisant la technique du "demand paging" est le choix de la page à attribuer : le nombre de pages, en effet, n'est pas illimité et il faudra faire un choix quant aux pages à invalider pour réattribuer cette mémoire physique. Une des techniques simples à utiliser est celle connue sous le nom "LRU" ou "least recently used".

Pour ce faire, on garde un historique de l'utilisation des pages mémoires et on réutilise systématiquement la page qui est restée le plus longtemps inutilisée. Cette technique nécessite donc une table contenant des index dans la table des pages triés dans l'ordre d'accès aux zones mémoires; or, pour des raisons d'efficacité, il est impensable de maintenir une telle table à jour, à moins de le faire par hardware, ce qui nécessiterait un matériel relativement complexe travaillant en parallèle avec l'unité centrale pour ne pas la ralentir.

Pour décrire un algorithme "LRU" légèrement modifié, nous allons admettre que le matériel utilisé a les fonctions décrites ci-dessus et nous allons faire l'organigramme.

Paging : "working set"

Une méthode légèrement différente de gérer la place mémoire disponible est celle connue sous le nom de "working set" : l'idée consiste à garder, par processus, les pages référencées indépendamment du time-slice attribué. Cela signifie que, avant de redonner le contrôle à un processus, on charge déjà son espace de travail en fonction des dernières pages utilisées lors du time-slice précédent. On évite ainsi une perte de temps en cours de time-slice par chargement de pages manquantes en cours d'exécution. Cette technique n'est intéressante que lorsque les pages du "working-set" peuvent être chargées pendant qu'un autre processus est en cours d'exécution. En utilisant cette même technique, on peut libérer la place d'un processus lorsqu'on lui retire le contrôle et par conséquent garder la description de son "working-set".

Paging : problèmes

Un cas "pathologique" du paging à la demande avec un algorithme LRU est l'exemple cité par Tannenbaum : supposons qu'un programme boucle en modifiant neuf pages de données, alors que son espace est de huit pages uniquement : par définition, l'algorithme fera choisir systématiquement comme page ancienne à éliminer la prochaine dont on aura besoin :

Paging : le "dirty bit"

Pour gagner en efficacité, il est utile de savoir si une page a été modifiée depuis qu'elle a été lue ou écrite sur disque. Si la copie mémoire est conforme, il n'est pas nécessaire de réécrire la page, et il suffit de l'invalider. On économise ainsi un transfert avec le stockage de masse.

Pour être utilisable, cela sous-entend que le hardware va indiquer quand une page a été modifiée; ce bit est nommé "dirty bit". On trouve une technique analogue dans la gestion des entrées-sorties disque avec la technique des buffers associés.

Réentrance

Lorsque, sur un système utilisé en time-sharing ou en multiprogrammation, plusieurs utilisateurs ont une activité identique, il peut être utile de réfléchir à la manière d'attribuer la place mémoire.

Une première règle à utiliser est donc de séparer systématiquement la partie programme d'une activité de ses variables et buffers. On verra alors pour une édition par exemple, la répartition de la mémoire comme suit :

soit, si le code du programme nécessite 10 K et les variables 8 K, un total de 90 K. Or la partie procédure (programme, représenté ici par PPP...) est identique pour les cinq utilisateurs, chargée en copie conforme en mémoire. On peut donc penser que, pour économiser la mémoire centrale, on ne pourrait la charger qu'une seule fois. On aurait donc :

soit, en ayant les mêmes tailles, une seule fois les 10 K du programme et cinq fois les 8 K de variables, soit en tout 50 K. On économise donc ici 40 K.

Les contraintes qui seront imposées par cette manière de faire seront principalement :

- séparation stricte du code exécutable et des constantes par rapport aux variables et buffers; l'utilisateur ne devra, en effet, que lire dans la partie programme, alors qu'il écrira et lira dans la partie variables,
- changement de contexte simple pour passer d'un utilisateur à un autre : la zone privée doit être adressée de la même manière avec les mêmes adresses pour tous les utilisateurs, au moyen d'un registre d'index, par changement de "bank" ou par modification de la table d'index dans un contexte de mémoire virtuelle etc.
- système de protection de la zone commune ou fiabilité du programme à toute épreuve ...

A noter que cette séparation est facile à faire lorsqu'on programme avec un langage de bas niveau tel que l'assembleur, mais nécessite un compilateur ayant prévu cette possibilité s'il s'agit d'un langage de haut niveau. De plus, l'éditeur de liens devra permettre cette séparation en plusieurs zones.

8.- Le time-sharing

Très souvent, on considère le time-sharing comme étant le partage du temps machine entre plusieurs utilisateurs dans un contexte de multiprogrammation multi-utilisateurs. En réalité, il s'agit du partage, non seulement du temps, mais encore de toutes les autres ressources d'un ordinateur travaillant dans ce contexte.

Dans ce chapitre, nous allons faire un survol des différentes ressources que l'on peut partager, et voir comment on peut le faire de manière plus ou moins efficace.

Partage des terminaux

Le partage des terminaux se fait de la manière la plus simple qui soit : l'utilisateur qui arrive à un terminal ou qui se connecte par le réseau est relié à un processus qui lui demande son identification et son mot de passe. Dès ce moment, le terminal est assigné et il le restera jusqu'à ce que l'utilisateur ait terminé ou que le système interrompe le travail pour une raison ou une autre.

Partage du temps machine

Le partage du temps machine est fait par attribution d'une tranche de temps à un utilisateur, et, lorsqu'elle est utilisée, il y a passage à l'utilisateur suivant. Une de ces tranches de temps est appelée un "time-slice".

Il est évident que si l'on veut que l'attribution du temps machine soit efficace, quelques règles doivent être respectées :

- l'utilisateur qui n'a pas besoin de temps machine ne reçoit pas le contrôle. C'est le cas des utilisateurs non identifiés ou en attente d'un ordre ou d'une donnée depuis le terminal,
- l'utilisateur qui a reçu le contrôle pour un time-slice donné et qui n'en a pas besoin entièrement, relâche le contrôle dès qu'il doit attendre un événement extérieur,
- l'utilisateur qui fait d'importantes sorties sur un terminal est automatiquement mis en attente, et le contrôle est passé au suivant,
- le choix du prochain utilisateur peut être fait de plusieurs manières, l'une d'elle étant de faire le tour de tous les utilisateurs de manière circulaire et d'attribuer du temps à ceux qui en demandent. Cette technique est appelée "round-robin",
- une autre manière d'attribuer le temps est de garder dans une queue de type FIFO les requêtes en temps machine, et d'attribuer un time-slice à celui qui est en tête de liste,

- une autre méthode, issue des deux précédentes, consiste à considérer deux types d'utilisateurs : ceux qui sont en attente suite à une requête en entrées-sorties, et ceux qui sont en attente après avoir "mangé" un time-slice complet; les premiers seront mis dans une queue FIFO ou circulaire rapide, avec éventuellement un time-slice plus petit, alors que ceux qui ont utilisé beaucoup de temps sont dans une queue lente. Dans ce cas, le système aura avantage à traiter tous les utilisateurs "interactifs" avant de passer aux mangeurs de temps machine : il donnera ainsi l'impression d'être rapide avec des temps de réponse courts. Ceux, par contre, qui demandent beaucoup de temps machine, attendront un peu plus longtemps, et, pour éviter de devoir faire beaucoup de changements de contexte, on pourra leur donner des time-slices plus grands,
- une quatrième méthode peut consister à donner une priorité de base à l'utilisateur, et l'on fait varier cette priorité suivant ses demandes en ressources : plus il en demande, plus sa priorité sera baissée, moins il en demande, meilleure elle sera.

Il est clair que ces méthodes ont toutes leurs avantages et leurs inconvénients; de plus, certaines implémentations sont à cheval entre plusieurs d'entre elles.

Le partage de la mémoire

Le partage de la mémoire sera fait selon l'une ou l'autre des méthodes décrites au chapitre "gestion de la mémoire". Il est bien clair que, dans la mesure du possible, les zones communes à plus d'un utilisateur devront être partagées, afin d'avoir un minimum de mémoire occupée à la fois.

Le partage de l'accès au disque

Les ressources disques seront partagées en utilisant le système de gestion de fichiers.

Afin d'éviter des ennuis, le système de gestion de fichiers sera intégré au système d'exploitation et ne sera pas à disposition de tout un chacun, si ce n'est en passant par des primitives. Ces dernières vérifieront que l'utilisateur a bien le droit d'utiliser tel ou tel fichier, et, si cela ne devait pas être le cas, elles lui retourneront un message d'erreur circonstancié.

Pour accélérer l'accès aux fichiers, on utilisera en général une technique qui consiste à avoir des tampons d'entrée-sortie disque au niveau du système d'exploitation et à minimiser les accès réels aux périphériques. Cette technique est connue sous le nom de technique des "tampons associatifs" ou "associative buffers". L'utilisateur ira donc lire ou écrire dans ce tampon qui est protégé et le système d'exploitation fera le transfert réel de manière regroupée, c'est-à-dire une seule fois par enregistrement physique, pour autant, bien sûr, qu'il dispose d'assez de tampons.

Afin de savoir si une réécriture du tampon sur disque est nécessaire, le système maintiendra un drapeau indiquant si la copie qui se trouve sur disque est conforme. Ce bit est parfois appelé "dirty bit", ou bit sale; s'il est positionné, une réécriture doit être faite, sinon, cela n'est pas nécessaire.

De plus, si une technique de pagination est utilisée sur le même disque, il faudra partager les accès dans le temps au périphérique. Il faudra donc répartir la charge en tenant compte des taux de transfert : le nombre moyen d'accès à un disque de technologie Winchester est d'environ trente accès à la seconde; le nombre de requêtes de transferts de pagination plus celui des requêtes de transferts du file-system ne devrait donc pas dépasser trente à la seconde, faute de quoi le processeur se mettra en attente d'entrées-sorties et perdra de son efficacité.

Certains constructeurs pensent que la limite du nombre de page-faults à la seconde ne devrait pas dépasser 5 à 6, d'autres vont jusqu'à 10 ou 12 au maximum, et ce pour garder un taux d'activité satisfaisant du processeur.

Une autre méthode pourrait être de mettre un second disque avec un contrôleur séparé pour pouvoir doubler le nombre d'accès. Dans ce cas, on essaiera de séparer la pagination du file-system.

Le partage de la place disque

Le partage de la place disponible sur disque peut poser de sérieux problèmes dès que de nombreux utilisateurs doivent cohabiter.

La technique la plus simple consiste à limiter la place que chaque utilisateur peut occuper : c'est ce que nous appellerons des "quotas disque". Pour faciliter la gestion, deux notions peuvent être utilisée :

- la notion de fichier temporaire, ou fichier de travail. Ce type de fichiers reste à disposition de l'utilisateur pendant la durée de la session, et, lors de la procédure de fin de session, un programme les détruit. Sont à considérer comme fichiers temporaires, à moins qu'ils ne soient déclarés différemment, tous les fichiers résultant de compilation (fichier listing ou objet), les fichiers de travail, tels que fichiers intermédiaires de tri, etc.

- la possibilité d'attribuer plus de place disque qu'il n'y en a réellement (quotas surestimés), en partant de l'idée que l'ensemble des utilisateurs ne sera pas au maximum des possibilités.

Peu de constructeurs offrent ces deux possibilités simultanément, en ayant, en général de bonnes raisons. Le choix est à faire en fonction des critères d'exploitation propres au site.

Un tel partage d'un disque peut poser des problèmes de temps d'accès, au file-system d'une part, à la pagination d'autre part. Comme vu au chapitre de l'accès au périphérique, le nombre d'accès peut, selon la charge, monter jusqu'à 20 à 30 par seconde. Avec beaucoup d'accès aléatoires, sans avoir de cache particulièrement grand, on peut optimiser les accès en utilisant la technique dite de l'ascenseur : on traite les demandes d'accès disque non pas dans l'ordre d'arrivée, mais dans l'ordre où les données se trouvent physiquement sur le disque en fonction de la position actuelle du bras. Cet ordonnancement des requêtes a permis, sur un système time-sharing, de passer de 20-22 accès par seconde à plus de 33 accès, soit environ 17 millisecondes de temps moyen d'accès.

Cette technique simple a toutefois ses limites. Pour ne pas prétériter des demandes de transfert pour des adresses en bord de disque ou au centre, il faut aller jusqu'au bout de chaque aller-retour. En outre, ce type d'optimisation devient particulièrement efficace lorsque beaucoup de processus indépendants les uns des autres demandent l'accès au disque et qu'ils peuvent sans autre être mis en attente.

Le partage des périphériques séquentiels

Les périphériques séquentiels ne seront pas partagés selon une méthode de time-slicing, pour des raisons évidentes.

Une méthode consisterait à demander à l'utilisateur de s'assigner le périphérique, puis qu'il l'utilise à sa guise. Cette méthode est peu efficace, car elle immobilisera le périphérique plus longtemps que nécessaire. La meilleure méthode, dans le cas d'une imprimante par exemple, est de différer dans le temps le transfert réel avec le périphérique physique en le simulant sur un support partageable rapide. Cette technique est connue sous le nom de spooling et est utilisée très souvent pour des lecteurs de cartes et des imprimantes.

L'utilisateur qui désire imprimer quelque chose écrira, de manière transparente, dans un fichier disque qui, une fois fermé, sera mis dans une file d'attente pour être pris en charge par le "despooler", qui est le programme qui vide la queue en imprimant ainsi à la vitesse maximum définie par le hardware. Ainsi, l'utilisateur n'attendra jamais de pouvoir disposer de l'imprimante, vu qu'il en a plusieurs, virtuelles il est vrai, à disposition.

Traitement par lot

Si, sur un système time-sharing, une possibilité de batch est implémentée, il faudra une gestion basée sur des files d'attentes pour traiter les utilisateurs travaillant en batch. Une telle queue peut très bien être prévue avec des niveaux de priorité différenciés et des possibilités de démarrage à une certaine date ou heure.

L'utilisateur fournira alors au système un fichier qui contiendra toutes les informations demandées par le processus (compilation, chargement, exécution, données à introduire en fonction des questions qui pourraient venir).

Il est bien clair que, dans un contexte de ce genre, l'utilisateur batch sera en fait un utilisateur time-sharing, soumis aux mêmes règles de protection, d'utilisation des ressources particulières etc., mais avec une gestion des time-slices différente. Ce mode sera surtout utilisé pour de grosses réorganisations et mises à jour de base de données, pour des backups, etc.

Sur certains systèmes où un pseudo time-sharing est à disposition, on dispose d'un éditeur et d'un gestionnaire de fichiers, et l'utilisateur peut lancer des exécutions et compilations en soumettant le travail en batch.

Traitement en temps réel

Si une machine permet le temps réel en exécution concurrente avec le time-sharing, il sera important de veiller à ce que les priorités soient bien définies : une erreur de gestion d'icelles peut empêcher une exécution correcte soit du time-sharing, soit du temps réel.

La création de tâches temps réel devra donc être réservée au responsable du système ou au moins à un utilisateur privilégié qui veillera à ne pas pénaliser un des modes de travail par rapport à l'autre.

Comptabilité des ressources utilisées

La comptabilité des ressources utilisées sur un système time-sharing est plus compliquée que sur un système batch. Si elle doit être tenue, il faudra tenir compte de tous les facteurs dont les plus importants sont :

- le temps machine réellement utilisé, si possible sans tenir compte des accès directs mémoire faits par des périphériques au profit d'autres utilisateurs,
- le temps de connexion, c'est-à-dire le temps où l'utilisateur a occupé une porte d'entrée sur le système et aura occupé un processus,
- le temps d'utilisation des périphériques, en particulier du disque,
- la quantité de mémoire bloquée, en fonction de la durée,

- les périphériques communs, tels que imprimante, plotter, lecteur et perforateur de cartes, unité de bandes magnétiques, etc.
- stockage disque permanent bloqué par l'utilisateur,

Certains constructeurs ne font pas le détail et donnent un coefficient à chacune des ressources. Ils ne comptabilisent que des unités de facturation, et, si une facturation réelle a lieu, le gestionnaire du système multipliera ces unités de compte par un facteur donné.

Il est possible d'organiser la comptabilité du système de manière à ce qu'elle débite au fur et à mesure l'utilisation des ressources utilisées, et que, si le budget est dépassé, lui interdise tout nouvel accès. L'autre méthode consiste à enregistrer dans un fichier toutes les opérations à comptabiliser, ce qui permet un contrôle après coup en cas de contestation, par exemple, mais qui a l'inconvénient de nécessiter une certaine place disque.

En conclusion, citons quelques avantages du time-sharing, tout en précisant que cette liste n'est pas exhaustive :

- regroupement en une puissance de calcul de plusieurs postes de travail, donc diminution des coûts pour une utilisation moyenne du matériel avec des pointes de forte utilisation,
- répartition selon les demandes de cette puissance de calcul,
- mise à disposition du temps machine non utilisé à d'autres utilisateurs,
- partage des utilitaires, compilateurs et autres logiciels qui seront en exemplaire unique sur disque,
- mise à jour des logiciels de base facilitée,
- communication de données facile d'un utilisateur à un autre, car sur le même système,
- méthodes de protections efficaces et adaptées sur les systèmes d'exploitation plus récents,
- facilité de mise en oeuvre de "backup" grâce à une comptabilisation meilleure de l'utilisation des fichiers.

Un système time-sharing peut très bien être intégré à un réseau qu'il soit local ou distant. Une méthode pourrait très bien être d'utiliser un système time-sharing dans un réseau local comme serveur disque ou comme puissance de calcul supplémentaire lorsque le micro-ordinateur devient insuffisant. Il n'est pas une alternative à un nombre de postes de travail indépendants, mais plutôt complémentaire avec les techniques de mise en réseau actuelles.

9.- Gestion du parallélisme

Certains traitements peuvent être programmés pour 2 ou plusieurs processus en parallèle. D'autres peuvent être décomposés pour être exécutés en parallèle afin d'accélérer le traitement global. Les lois de la physique donnent des raisons pour le parallélisme. D'après Einstein, les signaux électriques ne peuvent être transmis plus vite que la lumière (300.000 km/sec ou 1 pied/nsec). Si l'on considère l'exemple d'un CPU qui doit accéder à des données en mémoire à une distance de 1 pied, il lui faudra 1 nsec pour la demande et 1 nsec au moins pour la réponse. Un ordinateur qui possède plusieurs processeurs physiques pourra conduire plusieurs processus simultanément. C'est ainsi qu'un ordinateur comme le CRAY-1, considéré comme l'ordinateur le plus rapide au monde, peut exécuter des instructions plus rapidement que la vitesse de la lumière. En réalité, il s'agit d'une structure dite de "pipe-line" qui consiste à décomposer l'exécution d'une instruction en étapes élémentaires traitées en parallèle par plusieurs processeurs. De la sorte, le temps moyen d'exécution d'une instruction est considérablement accéléré.

Toutefois, tous les ordinateurs ne possèdent pas plusieurs processeurs capables de travailler en parallèle. Dans ces cas, une simulation du parallélisme s'impose : le temps est découpé en tranches et attribué à différents processus successivement. Voir le chapitre sur le time-sharing.

Le traitement de plus d'une tâche à la fois sur un ordinateur (parallélisme) peut se faire par une exécution concurrente des tâches (simultanéité apparente) ou par une exécution simultanée.

La simultanéité réelle ne peut être obtenue que par la prolifération des unités fonctionnelles. Elle implique la duplication (ou la multiplication) de certains éléments de la machine. L'exécution concurrente peut être réalisée soit par la machine elle-même, soit par le logiciel. L'effet est obtenu par multiplexage et repose sur les différences de vitesse d'opération entre différents composants du système.

Le multiplexage machine implique l'existence d'une famille d'unités à servir et d'une unité fournissant le service, l'unité de service ou contrôleur. Le contrôleur distribue à tour de rôle son attention sur toutes les unités, selon un algorithme implanté dans sa logique et tel que chacune recevant la quantité nécessaire d'attention au moment requis. Elles fonctionnent comme si elles recevaient un service continu de la part du contrôleur.

Le multiplexage par logiciel est à la base des systèmes à multiprogrammation et à temps partagé. Dans ces deux types de systèmes, on utilise un algorithme programmé pour partager la puissance de l'unité de traitement et la distribuer aux différents programmes en attente d'exécution.

Le taux de commutation et la vitesse d'exécution de l'unité centrale étant beaucoup plus élevés que le taux d'interaction entre chaque programme et son utilisateur au terminal par exemple, il est possible de donner l'illusion que plusieurs programmes sont exécutés en parallèle, comme s'il y avait un processeur pour chaque programme. En fait, le service est fourni de façon sérielle, par petits intervalles attribués de façon cyclique à chaque programme. Le parallélisme est donc du type exécution concurrentielle (simultanéité apparente).

Niveaux de parallélisme

Le fait que le parallélisme soit réel ou apparent dépend du niveau de la fonction à exécuter dans le système. Il existe plusieurs niveaux auxquels le parallélisme des opérations est possible :

- niveau du cheminement des données,
- entre les étapes de l'exécution des instructions,
- entre les instructions,
- entre les régions d'un programme,
- entre les tâches d'un programme structuré (multitasking),
- entre des programmes indépendants.

Cheminement des données

A ce niveau, le parallélisme est défini par la "largeur" des voies de transfert des données d'une unité fonctionnelle à l'autre à l'intérieur de la machine; c'est le nombre de bits qu'il est possible de transférer simultanément entre ces unités. En fait on transfère plus de caractères (ou de bits) à la fois.

Entre les étapes du traitement des instructions

A ce niveau, le parallélisme repose sur l'existence d'au moins deux étapes dans l'exécution d'instructions. Ces deux étapes sont habituellement constituées par une phase "instruction" où l'instruction est amenée de la mémoire à l'unité de traitement, puis décodée, suivie d'une phase d'"exécution" pendant laquelle l'instruction se déroule. Si les circuits logiques participant aux deux phases ne comportent pas la partie commune, partagée, alors la machine est capable d'exécuter une instruction pendant qu'elle prépare la suivante. Il en résulte un certain degré de superposition dans le traitement des instructions. Cette caractéristique est connue sous le nom de "look-ahead".

Potentiellement, cette caractéristique permet de doubler le taux de traitement de la machine. Mais en fait, ce doublement est rarement atteint à cause de goulets d'étranglements apparaissant dans le système : interférence d'accès à la mémoire retardant l'arrivée de l'information requise par l'une des phases, retards dus au traitement des instructions de branchement conditionnel qui empêchent l'arrivée de l'instruction suivante jusqu'au moment où la direction de branchement est connue, ou retards dus à la durée variable d'exécution de certaines instructions en fonction des opérandes (division, par exemple), ce qui immobilise l'unité d'exécution pendant une période plus ou moins longue et retarde donc le démarrage de l'instruction suivante.

Entre les instructions

Le parallélisme entre instructions est une caractéristique définie dans l'architecture de la machine et qui repose sur la prolifération de certaines unités fonctionnelles. Le parallélisme est d'ailleurs souvent limité à la phase d'exécution. Le CDC 7600, par exemple, possède une famille d'unités d'exécution spécialisées opérant indépendamment l'une de l'autre. L'unité de préparation d'instruction décode et distribue les différentes instructions aux unités d'exécution spécialisées. A un instant donné, on peut avoir par exemple une multiplication virgule flottante, un transfert à mémoire, une addition d'entier s'exécutant simultanément. Un résultat similaire peut être obtenu par la technique du "pipe-line", utilisée par l'IBM/360-195, par exemple. Le "pipe-line" est une extension de la technique "look-ahead". La phase d'exécution des instructions est fragmentée en plusieurs étapes intermédiaires. A un instant donné, il y a plusieurs instructions en train d'être exécutées, chacune à une étape différente de son exécution.

Entre régions d'un programme

Le parallélisme entre régions d'un même programme est obtenu par diverses combinaisons entre machine et logiciel. Pour qu'il y ait simultanéité réelle dans l'exécution des différentes régions, la machine doit être dotée de possibilités de parallélisme au niveau de la phase de décodage/préparation d'instructions : ce doit être une machine à courants d'instructions multiples. Par contre, la simultanéité apparente entre régions d'un programme peut être obtenue avec un logiciel de multiprogrammation. On appellera région une zone de code telle que son exécution peut avoir lieu indépendamment d'autres zones. Par exemple, deux zones constituent des régions si elles ne se passent pas de paramètre et si leur exécution peut être lancée de façon indépendante. les techniques machine/logiciel nécessaires pour une simultanéité réelle ou apparente entre deux régions impliquent l'existence d'un moyen de :

- reconnaître leur indépendance,
- démarrer leur exécution "en parallèle",
- établir les conditions d'une fusion ultérieure des deux chemins parallèles.

Entre tâches formalisées

Le parallélisme entre tâches formalisées d'un programme structuré est obtenu par l'intermédiaire du programme de contrôle du système (operating system) répondant à des directives données par le programmeur. Dans un système à multiprogrammation, une fonction a pour effet de placer une tâche fille (un programme demande l'exécution d'un sous-programme "fille") sur une file d'attente du dispatcher où elle va être servie, en même temps que la tâche mère, selon les algorithmes de dispatching du système. Dans les systèmes multiprocesseurs contemporains tels que : l'IBM 3033MP ou l'Univac 1110/80 où les unités de traitement se partagent la file d'attente du dispatcher, il est possible que la tâche mère et la tâche fille s'exécutent simultanément. Mais dans les deux cas, multiprogrammation ou multiprocessing, un certain nombre de fonctions de synchronisation sont nécessaires pour coordonner l'accès aux ressources partagées. La différence fondamentale entre parallélisme régional et parallélisme de tâches réside dans la dimension de l'unité à traiter et dans l'utilisation du programme de contrôle dans le deuxième cas.

Entre programmes indépendants

Le degré le plus général de parallélisme est obtenu avec la possibilité d'exécuter simultanément des flux de programmes entièrement indépendants. Il est réalisé par couplage plus ou moins lâche de systèmes indépendants en une association formant un système multiprocesseur. Le couplage peut être de différents types ou degrés :

- partage d'unités de contrôle et/ou d'unités E/S,
- partage des flux d'E/S des jobs (shared pool),
- gestion de l'ensemble du complexe par un des systèmes.

En dehors des contraintes normales de sérialisation dans l'utilisation des ressources partagées, unités physiques ou fichiers, les deux premiers arrangements ne requièrent que peu ou pas de coordination entre les différents flux de jobs, qui peuvent être attachés à l'un ou l'autre des systèmes dans le premier cas, ou fusionnés et traités en parallèle par les différents systèmes dans le deuxième cas. Dans le troisième cas, on introduit un gestionnaire afin de coordonner l'activité des systèmes du complexe.

Processus parallèles

Quand un programme est à exécuter, il doit être assimilé à un processus. Ce processus est caractérisé par son état : le program counter, le process-status-word, le stack pointer, les registres et par son espace d'adresses dans lequel le programme et les données peuvent être accédés.

Dans les systèmes d'exploitation simples, un nombre fixe de processus est créé au moment du bootstrap. Un processus nouveau attend dans une file jusqu'à ce qu'un processus libère une ressource.

Puis son espace d'adresses est chargé et enfin le programme est exécuté. Dans les systèmes d'exploitation plus sophistiqués, les processus peuvent être créés et détruits dynamiquement. On y trouve donc des instructions pour démarrer un nouveau processus. Le PL/1 est un langage de programmation qui permet d'écrire des programmes avec gestion de parallélisme et gestion de processus (citons aussi le Modula-2, le PASCAL concurrent, l'Epsimone, le C, le Portal, etc.).

Dans l'exemple, le mot TASK informe le compilateur PL/1 qu'un nouveau processus doit être créé. Certains systèmes d'exploitation permettent qu'un processus en contrôle un autre. On y trouve les possibilités d'arrêter, redémarrer, examiner ou détruire un processus.

Synchronisation

Considérons un exemple pour expliquer la nécessité du mécanisme de synchronisation entre processus. Deux processus indépendants doivent pouvoir communiquer entre eux au travers d'un buffer partagé en mémoire. Un processus, qui sera appelé producteur, calcule des nombres premiers et les range dans le buffer. L'autre processus, appelé consommateur, se chargera de l'impression des nombres. Les deux processus sont exécutés en parallèle à des vitesses différentes. Si le producteur constate que le buffer est plein, il s'arrête (sommeil). Plus tard, lorsque le consommateur aura ôté un nombre du buffer, il enverra un signal pour réveiller le processus producteur. De même, si le consommateur découvre que le buffer est vide, il s'arrête. Quand le producteur met un nombre dans le buffer, il devra réveiller le consommateur.

Considérons un exemple : un nombre reste dans le buffer à la position 21, le pointeur IN est sur le 22 et OUT sur le 21. Le consommateur imprime le nombre et les deux pointeurs ont la même valeur. Mais au moment où le consommateur teste les pointeurs, le producteur a trouvé un autre nombre premier. Il envoie donc un signal pour réveiller le consommateur. Mais le signal est perdu car l'autre processus va se mettre en sommeil. Les processus ne pourront plus se synchroniser. Pour éviter cette erreur fatale, il faudra pouvoir tester si un processus est actif ou non au moment de l'envoi d'un signal.

Synchronisation de processus utilisant des sémaphores

Deux solutions sont habituellement prises en compte pour la synchronisation de processus :

- première solution : chaque processus est équipé d'un "bit de sommeil-éveil". Si un signal d'éveil est envoyé à un processus qui tourne, son bit est mis. Si le processus s'arrête et le bit est mis, il repart immédiatement et le bit est remis à 0. Le bit d'attente mémorise les signaux superflus pour un emploi futur. Bien que cette méthode résolve les conditions d'avance des processus, il ne peut y avoir que deux processus qui s'exécutent concurremment. Une solution avec N bits est trop pénible.
- deuxième solution proposée par Dijkstra (1968) : l'utilisation de sémaphore.

Le sémaphore est une variable qui peut être positive ou nulle. Sur cette variable, deux opérations sont possibles : un UP qui est une incrémentation de un de la variable et un DOWN qui est une décrémentation de un de celle-ci. Si un DOWN est envoyé à un sémaphore strictement positif, on décrémente de un et le processus qui demande le DOWN continue. Si le sémaphore est nul, le DOWN ne peut s'effectuer et le processus qui envoie le DOWN s'arrête jusqu'à ce qu'un autre processus envoie un UP sur ce sémaphore (sommeil). Si le sémaphore est 0 et que l'autre processus reste à l'arrêt, alors dès que le sémaphore est mis à un, le processus arrêté peut repartir et compléter son DOWN, remettant le sémaphore à 0. Les deux processus repartent. Un UP sur un sémaphore différent de 0 l'augmente de un. Le sémaphore est donc essentiellement un compteur qui mémorise les sommeils-éveils pour un emploi futur de telle manière que ceux-ci ne soient pas perdus. La propriété essentielle des instructions sur les sémaphores est qu'une fois qu'un processus a initialisé une instruction sur un sémaphore, aucun autre processus ne peut accéder ce dernier et ce, jusqu'à ce que le premier ait terminé son instruction.

Considérons un exemple : soit un tournoi avec 20 équipes de volley divisé en dix jeux (processus), chacun se jouant sur son propre court et un grand panier (sémaphore) pour les balles. Il n'y a malheureusement que 7 balles. A chaque instant il y a entre 0 et 7 balles dans le panier (sémaphore = 0..7). Mettre une balle dans le panier revient à faire un UP et ôter une balle, un DOWN. Au départ, chaque court envoie un joueur au panier pour prendre une balle. 7 peuvent faire un DOWN, mais 3 sont forcés d'attendre (ne peuvent conclure un DOWN). Les jeux sont arrêtés temporairement. Eventuellement un jeu se termine et un joueur met une balle dans le panier (exécute un UP). Cette opération permet à un des 3 joueurs en attente de prendre la balle (complète son DOWN) permettant la poursuite de son match. Les autres joueurs restent arrêtés jusqu'à ce que des balles arrivent de nouveau.

Communications entre processus

Comment résoudre le cas du buffer partagé ou celui de variables à partager entre processus ? La possibilité de traiter la communication et la synchronisation de processus en parallèle existe : associer une "boîte aux lettres" (mailbox) à chaque processus. Les instructions seront alors de la forme SEND (UP plus envoi d'un message (entier, string)) et GET (DOWN plus lecture d'un message s'il y en a, sinon le processus s'arrête).

10.- Principaux systèmes d'exploitation

Les principaux systèmes d'exploitation que l'on peut trouver sur le marché seront brièvement décrits ici. Il ne s'agit que d'une liste où un choix a été fait. Pour avoir plus de détails, le lecteur se reportera à l'abondante documentation qui existe, et en particulier aux revues périodiques, où il n'y pas de mois sans qu'un nouvel OS ne soit décrit, commenté ou amélioré.

Précisons encore, avant de commencer cette énumération, que très souvent un système est appelé DOS, ou Disk Operating System, et que ces DOS sont rarement compatibles; ils ne permettent, en général, que de disposer d'un système de gestion de fichier et d'un analyseur de ligne de commande pour démarrer un programme.

CP/M

Le CP/M est un système d'exploitation développé par Digital Research Corporation et est prévu pour le processeur Intel 8080 et ses dérivés tels que le Z-80.

Son principal inconvénient est le fait qu'il est prévu pour tourner sur le processeur le moins performant de la série afin de pouvoir être général, donc certaines possibilités du hardware ne sont pas utilisées efficacement. De plus, CP/M est figé à un type de processeur, et le fait d'avoir introduit CP/M sur des ordinateurs tels que l'Apple II nécessite une simulation du hardware, voire même du hardware supplémentaire sous forme d'un second processeur qui est un Z-80 ou un Intel 8080.

CP/M se caractérise par le fait qu'il s'agit d'un système monoprocesseur, avec soumission des travaux dans un mode batch. Aucune gestion de mémoire n'y est inclue, et la gestion des fichiers permet leur utilisation en séquentiel et en accès direct, mais pas en séquentiel-indexé.

MP/M

MP/M est une version multipostes de CP/M et a les mêmes caractéristiques de base. Comme CP/M, les nouveaux drivers sont difficiles à inclure et la version existant actuellement, bien que d'une certaine popularité, n'a pas un grand avenir.

MP/86

Pas de différences sensibles, si ce n'est le changement de processeur par l'utilisation d'un processeur 16 bits, le 8086.

CPM/86

CPM/86 existe en deux versions, le CPM/86 et le concurrent CPM. C'est ce système d'exploitation qui tourne sur le PC d'IBM et ses principales caractéristiques sont :

- support d'un disque dur de type Winchester,
- possibilité de travail en temps réel,
- file system avec mots de passe,
- taille maximum de mémoire gérée : un mégabyte,
- travail en multitâches, mais utilisateur unique.

Les autres caractéristiques sont celles des systèmes décrits ci-dessus.

Turbodos

Sa caractéristique est de pouvoir remplacer, sans modification des programmes, un système d'exploitation CP/M, car il est décrit comme compatible avec celui-ci.

Il serait différent de CP/M par une conception plus modulaire qui devrait permettre une famille complète avec départ au monoposte batch et possibilité de travailler en multitâches, ce qui permettrait, par exemple, l'implémentation d'un spooler pour des sorties imprimante. De plus, la capacité utile des disques serait meilleure et la fiabilité du système de gestion de fichiers supérieure.

Turbodos, en outre, permettrait la connexion sur un réseau local sans modification du système.

MS/DOS

MS/DOS, développé par Microsoft est un système d'exploitation assez élémentaire qui a pour but d'interfacer les programmes avec le hardware, à savoir le CPU, la périphérie standard telle que disque, clavier, écran, imprimante. En fait partie également une librairie de commandes et utilitaires tels que l'éditeur (très élémentaire), un éditeur de liens et un dévermineur appelé DDT par exemple.

Le PC/DOS d'IBM n'a pas, ou que très peu, de différences avec le MS/DOS.

Bien que n'étant pas un système d'exploitation multitâches, MS/DOS permet l'écriture de routines d'interruption, et, par là, l'envoi d'impression en "background". L'analyseur de commande est facilement modifiable, car il s'agit d'un programme portant un nom particulier. Il suffit de le remplacer par un autre pour changer le contexte utilisateur.

La paramétrisation de MS/DOS permet de l'adapter à la configuration du microsystème et à l'application de manière relativement simple.

Actuellement (été 1986), MS/DOS est disponible sur Olivetti en version 2.11. Le passage sur MS/DOS 3.1 peut être fait, mais il peut être utile de savoir que le possesseur de disques de plus de 10 Mbytes doivent reformater leur disque lors du passage à la version 3. Il en est de même pour le PC/DOS IBM version 3 ou 4.

Multics

Multics est un système d'exploitation destiné essentiellement à des systèmes super-micro ou mini, voire des grandes machines. Développé au MIT (Massachusset's Institute of Technology), ses buts, décrits en 1965 déjà, étaient :

- accès par terminaux distants sans problème et utilisation normale comme depuis un poste local.
- un système intégré de gestion de fichiers,
- suffisamment de contrôles pour assurer un partage sélectif des ressources et fichiers,
- la possibilité de créer une structure hiérarchique tant pour l'accès aux fichiers que pour l'administration du système,
- la possibilité de gérer des utilisateurs de types différents, petits et grands,
- offrir différents environnements de travail au gré des désirs de l'utilisateur,
- une idée directrice pour plusieurs générations de hardware,
- environnement de programmation avec gestion de mémoire virtuelle segmentée,
- édition de liens dynamique,
- système de gestion des messages d'erreur uniforme pour tout le système,
- notion d'anneau de protection pour l'utilisateur, connue sous le nom de "protection's ring",
- interface de terminal standard garantissant l'indépendance vis-à-vis du type de terminal,
- jeu de caractères ASCII au niveau de tout le système,
- réseau intégré de télécommunication (Arpanet),
- possibilité de communication entre utilisateurs,
- système d'aide intégré ("help facility"),
- spoolers intégrés au système,
- backups complets et incrémentaux, c'est-à-dire que seuls les fichiers modifiés sont copiés,
- notion de projet, permettant des validations et la comptabilisation des ressources utilisées différenciées,
- possibilité pour l'utilisateur de savoir quelles sont les ressources utilisées, et ce à chaque commande.

Multics est utilisé depuis 1969 sur différents sites et a servi de ligne directrice pour plusieurs autres systèmes d'exploitation. Multics a donné ses principales caractéristiques à beaucoup de systèmes d'exploitation et les systèmes time-sharing d'aujourd'hui ont quasiment tous certains aspects de Multics.

Unix

Développé dans les laboratoires de la compagnie Bell, Unix est un système d'exploitation que l'on trouve actuellement sur plusieurs ordinateurs, principalement des micro-ordinateurs à 16 ou 32 bits et des mini-ordinateurs.

Unix est écrit en langage "C", et, à part un petit noyau, est implémentable sans grands efforts si l'on dispose de ce langage pour l'ordinateur cible choisi.

Il s'agit d'un système multitâches, multi-utilisateurs, prévu à l'origine pour des machines PDP 11 de Digital et qui a ensuite été adapté sur beaucoup d'autres systèmes.

Ses caractéristiques principales sont :

- multiprogrammation, éventuellement utilisable dans un contexte multitâches suivant les implémentations, et multi-utilisateurs,
- entrées-sorties banalisées, utilisables en librairies réentrantes,
- gestion des utilisateurs avec protections entre eux selon un principe analogue à Multics,
- système de gestion de fichiers hiérarchisé, permettant les accès séquentiels et directs,
- communications entre processus très faciles.

De l'avis de certains, Unix est le système d'exploitation de l'avenir et supplantera ses concurrents comme CP/M l'a fait pour les micro-ordinateurs huit bits.

En début d'année 1985, trois grandes firmes européennes produisant des microsystèmes ont signé un accord concernant la mise en commun d'expériences et portant le choix du système d'exploitation sur Unix.

11.- Les virus

Historique

On peut considérer Core War comme l'ancêtre des virus. Développé dans les années 60 par trois informaticiens, dont un dénommé Moris, ce jeu était en fait un jeu d'ingénieurs et opposait deux programmes avec des phases de destruction de l'adversaire et des phases de reconstruction. Un peu plus tard, le Docteur Frédéric Cohen a fait des expériences de propagation de programmes notamment sur des machines Unix, mais a dû arrêter rapidement, personne ne voulant en assumer la responsabilité.

Développé à l'origine au Pakistan, réalisé par les frères Alvi en 1986, Brain a été développé en premier lieu pour protéger les logiciels contre le piratage. Le texte d'identification, avec nom et adresse des auteurs, figurait en bonne place et en clair dans le virus, vu qu'il n'existait aucune législation concernant ces problèmes et que, par conséquent, il n'était pas nécessaire de camoufler les origines. Il a été fort diffusé au départ, notamment avec des logiciels copiés et vendu à bas prix aux étrangers.

Peu après, un virus, Mac Mag, peu méchant, affichant un message de paix a été développé par Charles Brando. Ce virus s'est retrouvé sur un pack master d'une société de software et a été diffusé à grande échelle. Une procédure pénale a été engagée et Charles Brando a été condamné.

En 1987, on a recensé cinq virus, Alameda, Jérusalem, Lehigh, Vendredi 13 et Vienne, et, en 1988, Agiplan, Cascade, Ping-Pong, Stoned et Vienne-B. Ce n'est qu'en 1989 que le nombre de virus a vraiment commencé à augmenter et en 1991 qu'il a pris l'ascenseur.

Monde Ms-Dos

année	nombre de virus
1986	1
1987	6
1988	11
1989	33
1990	255
1991	>1000
1992	>1800
1993 (sept)	>2150
1994 (sept)	>2900

Parmi ces virus, certains ont fait des ravages importants. On peut citer le ver internet. Il ne s'agit pas d'un virus à proprement parler, mais d'un programme destiné à saturer un réseau. En novembre 1988, un soir, des opérateurs ont constaté que la charge du réseau, plutôt que de baisser en fonction de l'heure, commençait à augmenter pour arriver à des charges de l'ordre de 90 % au bout de quelques heures et, dans la même nuit, arriver à toucher plus de 6000 machines qui ont dû être mises hors service. A l'origine de ce ver, un dénommé Moris (ce n'est pas le même que celui de Core War, mais son fils), qui a fait des essais sur internet et Unix.

En novembre 1989, trois virus différents connus sous le nom de Vendredi 13 ou Alerte Datacrime ont été diffusés. Fortement médiatisé, le problème a fait peur à plus d'un individu et beaucoup de journaux en ont tiré de quoi faire un maximum de copies.

Un autre virus, la disquette AIDS, présentant le type "cheval de Troie", a utilisé comme système de diffusion l'intérêt des personnes touchées. Envoyée à tous les participants d'un congrès médical concernant le SIDA ayant eu lieu en Norvège, l'auteur a joué sur l'intérêt et la motivation des destinataires pour la diffusion. Ce virus annonce qu'il s'est installé, met en garde l'utilisateur qu'il doit payer la licence pour le produit en indiquant un compte bancaire dans un pays tiers. Les polices de France, Belgique et Suisse notamment ont pu remonter la filière et ont découvert que c'était un professeur de Harward qui avait lancé ce virus. Toutefois, ce dernier avait pris la précaution de munir le logiciel d'une mise en garde interdisant l'usage du logiciel aux USA... il n'a pu être condamné pour ces raisons. Il a toutefois fini en maison de santé sous d'autres prétextes.

Frodo a été diffusé à large échelle (70'000 exemplaires environ) en mai 1991 sous forme de disquette encartée dans un périodique. Destiné d'abord comme logiciel antivirus (...), suivi d'une mise en garde et enfin, dans le numéro qui suivait, d'un correcteur, il a débouché sur une action pénale en vertu de la loi française de 1988. Actuellement (octobre 1991), la procédure n'est pas terminée, mais la justice a su s'entourer de spécialistes et trois inculpations ont eu lieu.

Enfin, la Suisse n'est pas en retard, puisque un virus de la troisième génération, particulièrement technique, a été découvert récemment : Tequila. D'origine helvétique, il a fait des ravages notamment au nord de la France et en Belgique.

Rappel concernant les risques informatiques

On peut diviser les risques informatiques en trois catégories : les risques accidentels, tels que feu, foudre, inondation etc., les erreurs involontaires telles que copie à l'envers, formatage de support par erreur, etc. et enfin la malveillance. C'est ce dernier cas que nous allons détailler quelque peu ici :

La malveillance, en informatique, peut se décomposer en plusieurs cas :

- Fraude : elle amène un gain, direct ou indirect, à son auteur ou à un tierce personne avec qui il est en contact.
- Détournement d'information : il est à assimiler à des violations du secret commercial ou industriel, et peut être réprimé comme tel.
- Détournement de logiciel : qui est communément appelé aussi "piratage" et qui crée un manque à gagner aux auteurs et distributeurs de ces logiciels.
- Destruction ou modification d'information, qui peut être manuelle (l'opérateur est présent et manipule) ou par le biais d'une infection informatique (virus).

C'est principalement de ce dernier cas que nous allons parler dans les lignes qui suivent.

Bombe logique

La bombe logique est un programme tout à fait normal auquel on a ajouté une fonction "bombe" intégrée dans l'exploitation usuelle, soit lors de sa conception, soit après-coup. Un exemple classique est celui du programmeur qui introduit une fonction de destruction des données si son nom n'apparaît plus dans la liste de paie pendant deux mois. Ce genre de procédé est souvent lié à des revendications salariales ou à un chantage.

Cheval de Troie

Le principe du cheval de Troie consiste à inclure, dans un programme spécifique, un fonction de que l'utilisateur ne voudrait pas exécuter (destruction par exemple). Un cheval de Troie n'a pas de fonction de duplication, et par conséquent il est important de le rendre attrayant, afin qu'il se diffuse largement. L'ensemble des chevaux de Troie est donc disjoint de celui des virus. L'exemple connu est le virus AIDS décrit en tête de ce chapitre.

A titre d'exemple, voici quelques virus entrant dans la catégorie cheval de Troie :

- CDIR.COM est un utilitaire qui affiche le répertoire en couleur... et qui détruit des secteurs du disque.
- ARC513.EXE est un utilitaire de la famille ARC qui sert à compacter des données... et à effacer le secteur zéro.
- NOTROJ.COM est un programme qui sert à détecter et supprimer les chevaux de Troie... et qui en est un lui-même.
- FLU4TXT.EXE est un antivirus qui est lui-même un virus.

Il convient donc d'être particulièrement méfiant avec ces petits utilitaires disponibles dans le cadre du marché "shareware" ou "freeware", qui sont des vecteurs préférentiels pour les virus.

Les vers

Les vers sont des programmes qui ne détruisent pas d'information, mais qui ont pour but de saturer soit les télécommunications, soit les supports magnétiques. Un exemple classique et souvent cité est cet employé d'un très grand constructeur qui, en décembre 1988, a envoyé ses voeux aux clients sous forme d'un dessin d'arbre de Noël et de la procédure permettant de faire de même... A noter qu'une manipulation humaine était nécessaire pour le faire continuer, mais en multipliant le nombre de destinataires.

Les virus et l'environnement technique

Il convient d'abord de rappeler qu'un virus peut être stocké de manière tout à fait inoffensive. Ce n'est que lorsque le code viral est exécuté qu'il fait courir un risque. En conséquence, un serveur ou un ordinateur central ne risque rien si le code exécuté sur la machine ne peut pas être modifié par les utilisateurs ou l'administrateur, même à son insu.

Le mode d'action consiste en général à détourner une ou plusieurs interruptions système, soit DOS, soit BIOS, pour en faire un traitement particulier.

Les spécialistes s'accordent à dire que, actuellement, aucun virus (mais pas les vers) n'a infecté d'ordinateur central, sauf lorsqu'il y a une partition et une émulation de micro-ordinateur, typiquement MS-DOS. Mais il ne faut pas oublier qu'un ordinateur central, ou un serveur, peut être un lieu de stockage de virus, voir un vecteur servant à le relayer plus loin. En outre, on a souvent tendance, notamment pour le passage par les circuits de télécommunications, à compacter les programmes et textes : il convient de ne pas oublier que les détecteurs ne peuvent pas trouver les virus sous leur forme compactée.

Mais voyons quelques caractéristiques des virus que l'on trouve sur le marché actuellement :

Les virus mutants

Certains virus sont modifiés quelque peu par des personnes plus ou moins bien intentionnées. Bien que de la même base et utilisant le même mécanisme, il n'en sont pas moins une variante. On a vu déjà revenir de nouvelle variante de virus connu après un certain temps.

Les virus automutants

Certains virus, pour être moins reconnaissables, utilisent un encryptage pour le code, respectivement les données dont ils ont besoin. Cet encryptage peut varier d'une génération à l'autre; il n'est pas rare de voir 16 ou même 256 variantes d'encryptage pour un code. Cette manière de faire complique passablement la détection par les logiciels usuels.

Les virus furtifs

Les virus furtifs sont des virus qui utilisent des techniques de camouflage pour ne pas être détecté par le simple utilisateur. La technique la plus classique consiste à modifier la taille du fichier disque lors de l'examen du répertoire pour donner l'illusion que rien n'a été modifié. Exemple : un virus se loge dans les fichiers de type .COM. Il rallonge ces fichiers d'une longueur de 3120 bytes. Il conviendra donc de reconnaître, lors de l'appel aux fonctions de recherche des caractéristiques des fichiers, ceux qui sont infectés, et de soustraire à la longueur la taille de 3120 bytes.

Second exemple : un virus se loge dans le secteur de bootstrap de la machine. Un utilisateur averti peut, au moyen d'un utilitaire simple de type Norton, aller examiner ce secteur. Pour ne pas être détecté, un tel virus peut reloger, par exemple dans un secteur "pseudo-défectueux", le vrai secteur de boot et rediriger les lecture disque sur ce secteur, camouflant ainsi la modification.

Les virus défensifs

Ce type de virus essaye de camoufler sa présence par des astuces diverses. Une des plus classiques est de contenir des empreintes de divers virus qu'il dissémine sur le disque, de telle manière à faire croire qu'il y a d'autres virus, restant, par là même, noyé dans la masse.

Les fonctions d'un virus

Les fonctions principales d'un virus sont d'abord la reproduction, le temps d'attente, le déclenchement et enfin le dommage.

Le cycle de vie qui en découle est le suivant :

- exécution du virus, avec chargement préalable en mémoire (tout code doit être en mémoire pour pouvoir être exécuté).
- test de déclenchement de reproduction et reproduction du code du virus.
- test de déclenchement du dommage, et exécution du dommage.

A noter que la fonction de reproduction et celle de dommage ne sont pas forcément liées, et peuvent avoir des temps différents. Un temps de latence permet de mieux camoufler l'infection et, par conséquent, de la propager plus.

Le code du virus peut se loger à différents endroits : on différenciera principalement les "virus système" qui vont se loger dans le secteur de boot, dans la table de partition, en dehors du disque (secteurs prétendus défectueux ou en dehors du disque logique), dans les deux fichiers systèmes cachés (IO.COM et MSDOS.COM, ou autre nom analogue selon la version du DOS) ou dans le COMMAND.COM.

Le second type de stockage du code du virus est dans les fichiers, de préférence les exécutables (.COM, .EXE), les drivers (.SYS appelés depuis le CONFIG.SYS) ou les overlays. Le code peut être ajouté au début, au milieu ou à la fin du programme exécutable, et ceci en recouvrant ou déplaçant le code original.

Le troisième type classique est le virus "compagnon". Il suffit, dans ce cas, de créer un programme d'extension .COM portant le même nom que le .EXE. Le code viral doit alors simplement, après avoir exécuté son action propre, lancer le .EXE. La stratégie de recherche des commandes du DOS lance, en effet, toujours le .COM s'il existe, et ne cherche le .EXE qu'après. Il suffit donc de cacher le fichier et le tour est joué.

Les générateur de virus

Afin que tout un chacun soit capable de créer un virus, une nouvelle gamme de produit a vu le jour : les générateurs de virus. Le but est bien sûr la multiplication des virus d'une part, mais aussi de multiplier le nombre de créateur de virus. Ci-dessous un certain nombre de ces programmes :

DAME : *Dark Avenger Mutation Engine*. Créé par Dark Avenger, auteur de Eddie, V2100, V1024, DAME est le seul générateur-$outil de virus. Il est d'origine bulgare. Il permet d'ajouter des fonctions supplémentaires à des virus existant, notamment la création d'un code auto-mutant rendant sa détection très difficile. Bien que très dangereux, il est relativement rare, car il faut disposer du code source du virus à modifier, disposer de DAME et disposer de bonnes notions d'assembleur.

VCS : *Virus Construction Set*. D'origine allemande, VCS est un logiciel qui génère des virus. Il est de danger limité, mais efficace : destruction de config.sys et d'autoexec.bat. VCS 2 est la version anglaise.

VCL : *Virus Construction Laboratory*. Il s'agit d'un véritable applicatif permettant la construction de virus.

GENVIRUS : *Générateur de Virus*. Ce logiciel, français d'origine, est distribué directement par son auteur. Pour éviter les problèmes dûs à la législation française, il est présenté comme un logiciel de test des anti-virus. Il permet la création de module de tests (à interpréter !!!) puisque son auteur écrit : "Genvirus a été écrit pour comprendre et démontrer le fonctionnement des virus informatiques et également pour tester l'efficacité de la protection des programmes antivirus disponibles sur le marché". Il propose un certain nombre d'actions, telles que blocage du clavier, du contrôleur de disque, un défaut du contrôleur vidéo ou de la mémoire, effacement d'un fichier ou de tous les .EXE ou .COM, voire de tous les fichiers, d'un piste ou d'un disque complet, de la FAT, du répertoire ou du secteur de Boot. Il permet de choisir le déclenchement, par exemple la date anniversaire du patron par exemple, ou par une action particulière au clavier.

Project XVIR et PS-MPC ou *Phalcon-Skism Mass Produced Code Generator*. Il s'agit là de deux autres produits nouveaux destinés au simple utilisateur pour fabriquer ses propres virus.

Les dommages en résultant

Les dommages crées par les virus peuvent être de plusieurs types. Il convient pourtant de rappeler que ce n'est que de manière très exceptionnelle que des dommages matériels ont lieu au niveau du PC.

Ces dommages peuvent être directs ou indirects, destructifs ou non, permanents ou temporaires. Nous allons en citer quelques-uns, le détail pouvant être trouvé dans la liste de Patricia Hoffman, diffusée régulièrement aux USA.

- Dégradation des performances du système. Le virus Jérusalem, dans ses différentes variantes, peut baisser la performance jusqu'à un facteur 10.
- Modification de l'aspect de l'écran. Ping-Pong fait se balader un point sur l'écran, comme une balle, et modifie parfois le caractère affiché à l'emplacement de la balle.
- Les caractères se mettent à tomber vers le bas de l'écran. C'est le cas de Cascade, de 1701 et 1704.
- Des effets sonores peuvent avoir lieu. Vaccine, Oropax, Yankee Doodle sont de cette catégorie. Berlioz date de mars 91 et vous joue la marche funèbre, tout en bloquant le PC... on attend 28 minutes ou on fait reset !
- Modification des fonctions du clavier : cela va des leds d'affichage d'état jusqu'au codage des touches.

Le virus Flip est un cas concernant la Suisse : il a, en effet, été écrit par deux étudiants suisses et les bruits les plus divers ont couru, au point que, faisant leur école de recrues, il n'encourraient aucun poursuite. Cette raison était, bien entendu, entièrement fausse, mais c'était bien sûr le manque de législation en matière de crime informatique qui les a protégé.

La nouvelle loi en préparation permettra de punir les auteurs de tels méfaits.

Quant au virus Flip, c'est un virus très technique qui attaque le secteur de bootstrap, la table de partition et les fichiers .EXE et .COM. Lors de la phase d'attaque, il détruit la structure disque.

- sortie de caractères farfelus sur les périphériques (LPT1 par exemple). C'est le cas de mix-1
- croisement de noms de fichiers : si l'algorithme est clair ou le nombre de fichiers restreint, vous pouvez réparer, sinon...
- destruction de secteurs, fichiers, etc.

Sécurisation d'un PC

La sécurisation d'un PC peut avoir lieu à différent niveau. Elle doit être faite à tous les niveaux et doit être respectée par tous les utilisateurs si l'on veut éviter des dégâts.

Niveau 1 : Précontamination

Il s'agit en fait là des mesures préventives usuelles. Cela va de la notion de backup fait régulièrement et avec un tournus des supports, par les consignes précises et surtout jusqu'à l'information. Les virus informatiques ne sont pas à considérer comme des maladies honteuses que l'on camoufle.

Ci-dessous un exemple de consignes utilisateur données par Jean-Claude Hoff sous forme de dix points :

1.- Protéger systématiquement contre l'écriture toutes les disquettes originales en apposant un sticker opaque sur l'encoche (disquettes $5^{1/4}$) ou en faisant apparaître le trou (disquette $3^{1/2}$).
2.- Faire de même pour les disquettes destinées à la lecture seule.
3.- Effectuer régulièrement une sauvegarde des fichiers de données et en conserver plusieurs générations, car l'une d'entre elles peut être endommagée, irrécupérable ou déjà contaminée.
4.- Examiner, ou faire examiner, régulièrement les disques durs pour vérifier qu'ils ne sont pas contaminés.
5.- En exploitation normale (hors opérations de décontamination), ne jamais faire démarrer un microordinateur avec une disquette, mais plutôt utiliser le système implanté sur le disque interne à l'ordinateur.

6.- Ne jamais transmettre ou utiliser des disquettes sans qu'elles aient été préalablement certifiées.
7.- En cas de symptômes confirmés de contamination, isoler immédiatement le mini-ordinateur : arrêter les traitements en cours, déconnecter l'ordinateur du réseau, ne plus utiliser le matériel, ne plus transmettre de disquettes, prévenir tous les utilisateurs concernés et prévenir, voire faire intervenir, les spécialistes internes ou externes.
8.- Toujours posséder en réserve une copie saine (disquette du constructeur ou certifiée saine) des logiciels du système d'exploitation.
9.- Toujours posséder une copie certifiée des utilitaires externes et des principaux programmes d'application.
10.- Transmettre sans délai toute disquette ou tout logiciel suspect vers les spécialistes internes ou externes.

Niveau 2 : Détection

La prévention commence toujours par un contrôle de la circulation des programmes dans un cadre réseau, et/ou des disquettes. Une certification préalable de tout matériel entrant dans un service (à l'état décompacté si nécessaire) permet de limiter les risques.

Il en va de même avec les disquettes sortant d'un service, et ceci pour des raisons de responsabilité.

Niveau 3 : Diagnostic

En cas de doute, et même régulièrement, il convient de faire un diagnostic en examinant tous les supports. Si un virus est détecté, il convient de l'identifier exactement afin de pouvoir mieux l'éliminer par la suite. Afin de pouvoir améliorer la lutte anti-virus, il convient de faire un copie des fichiers infectés et de la remettre (respectivement remettre la disquette) de tout nouveau virus aux spécialistes, internes ou externes. Cela permettra d'identifier de manière sûre, voir de refaire le test plus tard avec de nouveaux outils, afin d'améliorer la prévention.

Niveau 4 : Elimination

L'élimination du virus est à faire de manière très méthodique et en utilisant les outils appropriés, qu'il s'agisse d'outils DOS ou spécifiques. Il convient de garder en mémoire que le formatage est souvent une solution, mais qu'il existe des virus qui ne sont pas éliminés par cette action. Les spécialistes et les listes sauront expliquer exactement ce qu'il faut faire dans chaque cas.

Niveau 5 : Réparation

Il reste ensuite à réparer les dégâts crées par les virus. En principe, une bonne organisation des sauvegardes permet une récupération proche du 100 %

Outils de prévention

Les outils de préventions sont multiples et de qualité fort variée. Une technique souvent utilisée et efficace contre tous les virus consiste à utiliser une certaine redondance pour les fichiers et de la vérifier à chaque utilisation. Cette technique est lourde à implanter et ne va que pour des postes utilisant un nombre restreint et surtout très stable de logiciels. Entrent dans cette catégorie tous les logiciels de checksums, CRC etc.

Il existe également des logiciels qui vont tester régulièrement les vecteurs d'interruption, voire le contenu de la mémoire, pour constater un éventuel changement, et le signaler.

Lorsque de tels logiciels constatent une modification, il affichent en général un message plus ou moins circonstancié. Ce message n'est utile que s'il n'apparaît qu'en cas de problème. Les messages inutiles noient les messages importants et ont pour conséquence que l'utilisateur ne les lit plus.

Outils de diagnostic

Les outils de diagnostic sont multiples et variés. Peu sont sérieux et régulièrement mis à jour. Certains utilisent un nom porteur qui donne une impression de sérieux et ne détecte qu'un quart des virus existants, d'autres considèrent un virus mutant à 16 possibilités comme 16 virus différents... et augmentent ainsi le nombre de virus détectés. Il convient donc de n'utiliser que des outils sûrs et éprouvés dans le cadre de la recherche des virus, et surtout de n'utiliser que des versions à jour.

Conclusion

S'il fallait donner une définition du virus informatique, je dirais qu'elle ressemble étonnament à celle du virus au sens médical. Les deux ont les mêmes fonctions et ce qui les différencie, c'est leur nature. Un parallèle quasi constant peut être tiré entre eux.

Quant aux réactions des utilisateurs de PC, on constate que les nouveaux "infectés" passent par différents stades :

- incrédulité (cela n'arrive pas qu'aux autres !)
- intervention à chaud (souvent avec un temps limité, des conditions de stress, des utilisateurs tournant autour et sachant souvent mieux que les spécialistes ce qu'il convient de faire)
- mise en place de moyens limités

- mise en place de mesures de prévention
- réflexion
- mise en place de moyens homogènes et cohérents
- planification et
- intégration de la lutte anti-virus dans un plan global de sécurité informatique au sein du service ou de l'entreprise.

Les virus font désormais partie de l'environnement micro-informatique. On ne peut plus les ignorer et ce n'est qu'en informant d'une part, en supprimant la conotation de maladie honteuse que l'on arrivera à limiter leur propagation.

Les français ont abordé très rapidement le problème en légiférant. Une personne, une entreprise qui diffuse un virus, consciemment ou non, peut être condamnée jusqu'à trois ans de prison et/ou 2 millions d'amende. En Suisse, par contre, la législation et le code civil sont encore très pauvres sur le sujet : les seules condamnations qui ont lieu pour l'instant l'ont été sur la base de textes touchant à d'autres domaines et l'ont été par analogie : secret commercial, secret de fabrication ou industriel, droits d'auteurs etc. Le parlement fédéral planche toutefois sur le sujet et des premiers projets existent déjà. Quelle en sera leur teneur exacte et quand entreront-ils en vigueur ? Nous ne le savons pas encore aujourd'hui.

Quant au futur, ne nous risquons pas imaginer ce qui viendra. A court terme, on sait que des virus très techniques, utilisant plusieurs systèmes de propagation, directs et indirects, seront utilisés. Certains virus vont avoir une action que lors de rencontres (interaction entre virus), certains étant d'ores et déjà découverts. Une chose est certaine : ils seront de plus en plus sophistiqués et de plus en plus nombreux. Sachons donc travailler avec cette menace !

12.- Les transmissions

Le téléphone

En 1876, Alexander Bell inventa le téléphone. Deux ans plus tard, en 1878, la première liaison téléphonique fut ouverte, desservant 21 abonnés et contrôlée par un opérateur. Le processus d'établissement d'un appel était le suivant : l'appelant soulevait le combiné du socle, actionnant ainsi un contact de fermeture du circuit (boucle d'abonné) avec le central. Ce contact établissait un courant fournit par la batterie, qui actionnait un indicateur au central. L'opérateur ainsi alerté demandait quel était le destinataire voulu, puis appliquait un courant de sonnerie au destinataire. Lorsque le destinataire avait répondu, l'opérateur établissait le circuit au moyen de fiches, et notait l'heure pour pouvoir établir, par la suite, la facture. Lorsque les abonnés raccrochaient, l'opérateur voyait son indicateur tomber et il libérait le circuit, notant l'heure et la durée de la communication. C'était le début de la signalisation.

Les premières lignes interurbaine, reliant les centraux entre eux, datent de 1880. En 1892, Almon B. Strowger inventa le relais pas à pas. Les abonnés au téléphone pouvait ainsi établir eux-même les circuits en activant rapidement et à cadence fixe le contact de fermeture de circuit. La signalisation d'adressage est donc née. En 1896, Keith, Erikson et Erikson ont développé le cadran de numérotation pour générer les ouvertures et fermetures de boucle qui commandaient les relais pas à pas.

En 1924, les premières signalisation au niveau interurbain ont vu le jour. Vu les nombreuses interférences avec les tramways et trains, la signalisation interurbaine a passé du courant continu au courant alternatif à 50 Hertz. La signalisation entre abonnés et central d'une part, entre centraux d'autre part, se sont différenciées.

En 1928, Skillman proposa d'utiliser un codage multifréquences (MFC) pour la signalisation. Mais ce concept n'a été utilisé que dans le milieu des années 1940. Le DTMF (dual tone multi frequence) a remplacé le MFC par la suite.

En 1962, les premières lignes d'abonnés interurbaines numériques MIC TDM (multiplexage temporel) furent installées. Bien entendu, deux tendances émergèrent : le système américain à 24 voies, transmettant à 1,544 Mbits par seconde, et le système européen à 30 voies, transmettant à 2,048 Mbits par seconde.

Ce n'est qu'au milieu des années 1970 que le système entièrement numérique a vu le jour (système CCS ou common-chanel-signaling).

Normalisation

Les communications entre hommes sont de plus en plus importantes de nos jours et les moyens utilisés de plus en plus diversifiés. Plusieurs supports différents sont utilisés et chaque fournisseur ayant ses propres caractéristiques, une tour de Babel a été ainsi construite.

Constatant l'anarchie existante, et afin de ne pas favoriser l'un ou l'autre des constructeurs, des organismes de normalisation ont vu le jour dont nous citerons :

- L'ISO, ou International Standard Organization, qui regroupe plus de quatre-vingts instituts nationaux de normalisation, dans toutes les branches, et qui a pour but d'édicter des standard afin de favoriser les échanges internationaux,
- Le CCITT, ou Comité Consultatif International Télégraphique et Téléphonique, qui dépend de l'UIT (Union Internationale des Télécommunications), et qui publie des avis dans différentes séries telles que D, E, F (tarifs et régulations), R, S, U (télex, télétexte, vidéotex), T (fac-similé), V (transmissions de données analogiques) et X (transmissions de données numériques). Les membres du CCITT sont en principe les administrations postales des différents pays,
- L'ECMA, ou European Computer Manufacturer Association, qui publie également des standard dans la branche informatique et bureautique.

Ces différentes associations ont émis des descriptions des réseaux de transmission de données que nous ne décrivons ici que partiellement, vu l'étendue que pourrait prendre un tel document s'il voulait être exhaustif.

La transmission à distance : les modems

Pour pouvoir transmettre à distance de manière économique, il est évident qu'il faut utiliser une ligne sur laquelle les informations seront passées en mode série. Il serait, en effet, aberrant de commander une imprimante distante en mode parallèle qui nécessite au moins une dizaine de fils. Pour utiliser des lignes d'une certaine longueur, sans pertes de niveau, il est indispensable de coder les informations. Le réseau téléphonique a été, à l'origine, prévu pour transmettre des fréquences dans la gamme de l'audible, plus précisément entre 200 et 3500 Hertz environ. Il faudra donc convertir les signaux digitaux en signaux analogiques pour pouvoir les transmettre à travers les centraux téléphoniques, et, de plus, le niveau ne pourra pas être utilisé pour différencier les données, vu qu'il existe des amplificateurs par lesquels certains circuits passeront.

Pour coder les informations digitales en informations analogiques, on utilisera un modulateur et un démodulateur pour le décodage des informations analogiques en informations digitales. L'appareil qui effectue ces conversions s'appelle modem, mot tiré de MOdulateur - DEModulateur. Le CCITT a défini les caractéristiques des modems du côté équipement informatique dans l'avis V24 et, du côté du réseau téléphonique, dans les avis V21 pour des transmissions de 75 à 300 bauds, dans l'avis V23 pour les transmissions jusqu'à 1200 bauds et dans l'avis V28 pour celles jusqu'à 2400 bauds.

Pour les lignes louées (circuits téléphoniques loués à l'année, sans passage par des sélecteurs dans les centraux), une technique souvent utilisée est celle de la bande de base (base band) qui permet d'avoir des modems simples avec une seule fréquence et utilisant tout le canal. Ce type de codage n'est toutefois utilisable que dans le réseau local, à des distances inférieures, dans les conditions normales, à dix kilomètres environ.

Transmissions mono ou bidirectionnelles

Les transmissions peuvent être de trois types :

- monodirectionnelles, c'est-à-dire que le flux de données ne va que dans un seul sens. Ce type de transmission est utilisé en télex, à la radio et à la télévision, par exemple. Ce mode se nomme "simplex". Il présente l'inconvénient de ne pas savoir si tout a été reçu par le destinataire sans erreur.
- bidirectionnelles alternées, c'est-à-dire une transmission bidirectionnelle, mais en alterné. L'exemple le plus typique est la conversation entre radio-amateurs ou par "talkie/walkies" où chacun parle lorsque le correspondant a terminé. L'utilisateur est à l'écoute et il doit couper l'écoute s'il désire parler. Cette technique portera le nom de "half duplex". Par rapport aux transmissions simplex, il est nécessaire de disposer de transmetteurs et récepteurs aux deux bouts du circuit et les amplificateurs ou relais devront également être bidirectionnels.
- bidirectionnelles, c'est-à-dire que les deux correspondants sont capables de parler et écouter simultanément. Comme exemple que tout un chacun connaît, citons le téléphone, où aucune priorité ou alternance n'est nécessaire entre les deux correspondants. Cette technique nécessite l'utilisation de deux voies de transmission, qui peuvent éventuellement être multiplexées sur un seul et même support physique. Le terme utilisé pour désigner ce genre de circuit est "full duplex".

Le multiplexage

Il arrive souvent que l'on ait plusieurs terminaux situés dans un même bâtiment qui sont connectés sur le même centre de calcul. Il faudrait donc, théoriquement, une ligne de communication par terminal. Cette manière de faire revient vite chère en coûts de communication et c'est la raison pour laquelle on multiplexera les données sur une seule ligne de transmission.

Le multiplexeur devra donc se débrouiller pour transmettre à son homologue non seulement les données, mais encore des informations permettant d'associer à l'information une destination. Il serait, en effet, désagréable de recevoir des bribes de messages destinés au voisin.

Ce multiplexage peut se faire selon plusieurs techniques différentes :

Multiplexage de fréquences

On pourrait envoyer sur un même lien physique les données de plusieurs circuits en les modulant à des fréquences différenciées et en utilisant, avant le démodulateur, des filtres. Cette technique est utilisée entre centraux téléphoniques par exemple et pose un problème, à savoir la nécessité de disposer d'une large bande. Or actuellement, les PTT suisses offrent, avec garantie de qualité de transmission, au maximum de l'ordre de 15 KHz sur tout le réseau en lignes louées. Pour des fréquences plus élevées, les PTT offrent des modems et ne garantissent pas, à la commande, la possibilité de transmission à plus de 9,6 Kbauds. On peut en conclure que la bande passante garantie est de 300 à 15'000 Hertz environ, toute fréquence supérieure n'étant utilisable qu'après choix et test des lignes. Le multiplexage de fréquences n'est donc pas judicieux pour un utilisateur final actuellement.

Multiplexage temporel

Le multiplexage temporel consiste à allouer à chaque terminal un temps donné, et ce à chacun, l'un après l'autre. Un calcul simple permet donc de dire que, pour une ligne de 9600 bauds, on peut avoir deux terminaux à 4800, 4 à 2400, 8 à 1200 bauds, etc. Toutefois, le calcul n'est pas aussi simple : il ne faut, en effet, pas oublier qu'un temps de transmission des informations de contrôle entre les multiplexeurs est également nécessaire.

Multiplexage statistique

Le multiplexage statistique permet de connecter plusieurs terminaux à des vitesses dont la somme est plus grande que celle de la ligne principale. L'idée qui se cache derrière cette technique est que, à aucun moment, tous les terminaux demanderont des transmissions à la vitesse nominale. Seul un test dans la réalité permettra de dire si une ligne est utilisée de manière faible, moyenne ou est, au contraire, saturée. Dans tous les cas, il est préférable de disposer d'un système de contrôle de flux pour éviter, dans un contexte de multiplexage statistique, de perdre de l'information.

Sélecteurs de lignes

Si des terminaux sont répartis dans un bâtiment et ne sont pas utilisés tous simultanément, il est possible d'installer un sélecteur de lignes. Le sélecteur le plus connu, dans le domaine téléphonique, est le central d'abonné qui peut avoir beaucoup de lignes internes, et un nombre restreint de lignes externes. L'utilisateur qui désire obtenir une ligne externe doit la demander, et, si elles sont toutes occupées, il devra attendre la libération de l'une d'entre elles. Citons, dans le domaine informatique, deux marques de sélecteurs de ce type, installés soit au CERN, soit à l'université de Genève : Gandalf et Micom.

Technique du "polling"

Il arrive aussi que l'on installe plusieurs terminaux sur le même canal de transmission et que l'ordinateur central qui commande le circuit de transmission ajoute une adresse permettant de différencier les destinataires entre eux. Le terminal n'affichera donc que les messages qui portent sa propre adresse et ignorera les autres. Quant au passage du terminal à l'ordinateur, il aura lieu "à la demande", c'est-à-dire que le terminal stockera le message et ne l'enverra qu'à la requête de l'ordinateur. Cette technique permet l'utilisation d'un interface et d'une ligne pour plusieurs terminaux et la saturation de la ligne dépendra du temps d'attente que l'utilisateur admettra comme acceptable.

Le terme polling vient du fait que seul le maître de la ligne commande, et, en envoyant des "polls" aux terminaux, leur permet tour à tour de mettre des informations sur la ligne.

Les codes de caractères

Chaque système a un ensemble de caractères qu'il peut lire ou écrire. Au minimum, cet ensemble devrait comporter les 26 lettres de l'alphabet (majuscules), les 10 chiffres (digits) et quelques signes de ponctuation tels que virgule, point, point-virgule, deux-points, signe plus, signe moins, etc. Un jeu de caractères plus sophistiqué pourrait comprendre en plus, les lettres minuscules, un assortiment plus important de marques de ponctuation, une collection de caractères utiles en mathématique et en économie et même les lettres grecques.

Afin de transférer ces caractères dans un ordinateur, chaque lettre doit correspondre à un nombre, par exemple a=1, b=2, etc. La correspondance entre une lettre et un nombre est appelé code de caractère. Actuellement, les ordinateurs utilisent des codes de 6, 7, 8 ou 9 bits. Un code de 6 bits n'autorise la manipulation que de 2^6 = 64 caractères. Pour la plupart des applications, 64 caractères ne suffisent pas et les codes tiennent sur 7 ou 8 bits. Un code sur 7 bits autorise 128 caractères. Le code le plus répandu est le code ASCII (American Standard Code for Information Interchange). Le code à 8 bits le plus communément utilisé était le code EBCDIC d'IBM (Cf. table en annexe), qui maintenant a été remplacé par le code ASCII étendu.

La standardisation est particulièrement importante dans le domaine des télécommunications, car c'est le seul moyen par lequel il est relativement facile de transmettre de l'information. Le code ASCII est, actuellement, le code le plus utilisé dans le domaine de la micro-informatique et de l'informatique distribuée. Toutefois, ce code n'ayant que 128 caractères, et les applications en requérant de plus en plus, on l'utilise souvent avec une extension à 256 possibilités de représentation. (Citons, à titre d'exemple, le problème des accents que l'on retrouve en particulier dans la langue française). La standardisation n'est pas poussée encore très loin dans ce domaine, et la plupart des constructeurs utilisent leur propre code pour les 128 caractères supplémentaires. Ils nomment souvent ce code le "code ASCII étendu".

Protocoles de transmissions

Afin de faciliter le travail, la connexion entre ordinateurs, le multiplexage et bien d'autres fonctions encore, des protocoles de niveau plus ou moins élevés ont vu le jour. La normalisation de ces protocoles est une tâche ardue que les constructeurs et les organismes de normalisation n'ont pas encore résolu.

Il faut toutefois reconnaître que l'adaptation à un protocole donné peut nécessiter une certaine charge de travail pour gérer les communications. D'autre part, la récupération en cas de transmission altérée peut être lourde à gérer et le débit d'une ligne dépendra beaucoup des temps morts se trouvant entre les messages.

Afin d'expliciter un peu ces notions, nous allons détailler, dans les paragraphes qui suivent, les processeurs spécialisés et les problèmes de contrôle d'intégrité des transmissions.

Processeurs d'entrées-sorties

Les processeurs d'entrées-sorties doivent pouvoir transmettre de grandes quantités de données dans un temps très court. Si le CPU doit se charger de traiter les caractères les uns après les autres, une grande part de l'efficacité de ce dernier est perdue. Pour éviter le partage du CPU dans le traitement des entrées-sorties, la plupart des systèmes sont composés de processeurs spécialisés pour ce traitement. Ces derniers peuvent travailler en parallèle avec le CPU. On appelle parfois ces unités, des canaux (channels) ou unités de traitement périphérique (PPU, Peripheral Processing Unit).

Les codes de correction

Les ordinateurs, comme les êtres humains, sont enclins à l'erreur occasionnellement. Les unités d'entrées-sorties font appel habituellement à un déplacement physique des données et des erreurs sont probables. Ces erreurs peuvent être provoquées par de la poussière sur des têtes de lecture, des parasites sur des lignes téléphoniques, des chutes de tensions, etc. Ce chapitre traite des codes de caractères permettant de détecter la plupart des erreurs et même parfois de les corriger.

Contrôle au niveau caractère

Une méthode simple, et très utilisée, consiste à détecter des erreurs simples (un bit changé) en additionnant un bit de parité à chaque caractère. Dans le cas d'un code de parité impaire, le bit de parité est choisi de sorte que le nombre de bits à 1 dans le caractère, y compris le bit de parité, soit un nombre impair. Dans le cas d'un code de parité paire, le bit de parité est choisi de telle sorte que le nombre de bits à 1 soit un nombre pair. Considérons l'exemple suivant :

Une autre méthode consiste à déterminer un code de sorte que non seulement les erreurs sont détectées, mais une correction peut y être apportée. La méthode porte le nom de son auteur : Richard Hamming (1950). Dans un code de Hamming, on ajoute au caractère sur N bits K bits de parité et on obtient ainsi des caractères de longueur N+K bits. Les bits sont numérotés à partir de 1 (et non 0) avec le bit 1 comme étant le bit le plus à gauche..i.Hamming

Tous les bits dont le numéro est une puissance de 2 sont des bits de parité et les bits restants sont utilisés pour les données.

Contrôle au niveau message

Au paragraphe précédent, nous avons vu comment faire un contrôle au niveau du caractère que nous appellerons parfois aussi parité horizontale. On peut faire le même type de contrôle au niveau d'un message composé de plusieurs caractères, avec la même technique des nombres de bits pairs ou impairs. Une telle somme de contrôle porte en général le nom de block check character (BCC) et se calcule par des opérations "ou" exclusif sur toutes les données.

L'inconvénient de cette méthode est de ne pas détecter la perte ou l'adjonction de deux caractères identiques et, même associée à la parité horizontale, cette parité verticale est relativement peu fiable. C'est pourquoi une technique plus complexe et plus fiable est souvent utilisée, surtout lorsque les caractères transmis sont à huit bits, sans parité : la technique du CRC (cyclic redundancy check), parfois aussi appelé FCS (frame check sequence).

Le code de redondance cyclique est composé d'un nombre de bits équivalent à la puissance du polynôme générateur, en général 16, parfois 24. On obtient le CRC en multipliant la donnée (les caractères transmis) par la plus haute puissance du polynôme générateur, puis en le divisant par le polynôme générateur. On ne garde alors que le reste de la division et on l'additionne au CRC précédemment obtenu. Des démonstrations mathématiques que nous ne ferons pas ici permettent de montrer que seuls deux CRC de 16 bits sont fiables :

Le CRC-CCITT est en général celui qui est implémenté. Un algorithme très simple pour le calculer au fur et à mesure du décalage des bits dans l'interface de transmission fait que, dans la plupart des circuits intégrés de communication série synchrone, le contrôle est fait par hardware.

En principe, toutes les erreurs jusqu'à 16 bits sont détectées et 99 % des erreurs de plus de 16 bits également.

Pour plus de détails, le lecteur se reportera à la littérature, en particulier au manuel de John E. McNamara.

Mode de transmissions des caractères

Il existe trois types de transmissions de données couramment utilisés : les modes asynchrone, synchrone et HDLC. Ces modes ont chacun leurs avantages et inconvénients que nous allons décrire ici :

Transmission asynchrone

La transmission asynchrone est une transmission caractère par caractère, sans structure plus élaborée. Chaque caractère est précédé, sur la ligne, par un bit de début de transmission, appelé start-bit. Puis viennent les huit bits de données (bit de parité inclut s'il y a lieu) et enfin une séquence d'un, un et demi ou deux bits de fin, appelés stop-bit. (On obtient un stop-bit et demi en laissant le signal correspondant pendant une fois et demie la durée de transmission d'un bit sur la ligne. C'est, en effet, un des seuls endroits, voire le seul, où on parle de demi-bit).

On constate donc que chaque caractère est transmis avec un "overhead" de 2 à 3 bits, soit de 25 à 30 % ! Ce mode de faire devient donc très vite coûteux et c'est la raison pour laquelle il n'est en général utilisé que pour des terminaux travaillant en mode caractère. Les liaisons entre ordinateurs seront en général établies en synchrone ou en HDLC.

Transmission synchrone

Afin d'éviter la perte de temps de transmission des start-bits et stop-bits, en particulier dans les transmissions d'ordinateur à ordinateur, on utilise souvent une ligne synchrone : l'unité de transmission est le message que l'on fera précéder d'un ou plusieurs caractères de synchronisation, suivis des données à raison de huit bits par caractère, sans start-bit, ni stop-bit.

A partir du moment où la ligne est synchronisée, le débit des caractères doit être tenu, chaque byte étant formé de huit bits et le suivant commençant directement après le dernier bit transmis. Si l'ordinateur transmetteur ne tient pas le débit, on parlera d'"underrun" (sous-vitesse) et si c'est l'ordinateur récepteur qui perd un caractère pour des raisons de débit, on parlera d'"overrun" (sur-vitesse). Ce débit devra être tenu jusqu'à la fin du message, reconnue par un caractère ou une séquence de caractères donnés. Il est donc impensable de transmettre des messages sans disposer d'un protocole de transmission.

Le mode "bisync" ou BSC est un mode de transmission synchrone où la synchronisation entre l'émetteur et le récepteur se fait par au moins deux caractères de synchronisation.

HDLC - High level Data Link Control

Ce protocole a été défini par les instituts de normalisation et a été repris par les constructeurs, en particulier par Univac (UDLC) et IBM (SDLC). Il permet la transmission d'une chaîne de bits dans un contexte de trame appelé "frame" en anglais et qui correspond à un message. La transmission se fait également en synchronisme, ce qui signifie que le débit doit être tenu pendant toute la durée de la transmission d'une trame et que les erreurs d'overrun ou d'underrun peuvent également se produire.

Le début de trame est indiqué par un flag composé d'un bit à zéro, de six bits à un et d'un bit à zéro. Puis vient une séquence d'identification du destinataire et de contrôle, puis les informations et enfin le CRC. La trame est terminée par un flag, qui ouvre éventuellement de suite la trame suivante. Lorsque la ligne de transmission est au repos, il y a envoi continuel de flags, ce qui permet de garder la ligne synchronisée.

Afin de ne pas pouvoir perturber l'envoi d'une trame par une donnée contenant, par exemple, six bits à un, le transmetteur intercalera, après chaque cinquième bit à un, un bit à zéro, qui sera enlevé par le récepteur. Une trame composée de données dont tous les bits sont à un verra donc son temps de transmission augmenté de 20 % environ !

13.- Les télécommunications

Dans ce chapitre, nous allons traiter principalement des standard de télécommunication qui existent pour les transmissions de données. Nous parlerons surtout des réseaux publics. Les réseaux locaux seront traités dans un autre chapitre.

Le modèle ISO

Afin de standardiser quelque peu les télécommunications entre ordinateurs, l'ISO a proposé un modèle de protocole de télécommunication à plusieurs niveaux, nommé modèle ISO OSI. Le CCITT a défini, dans ses Avis, les protocoles à plusieurs niveaux, mais l'Avis X.25 ne définit que les trois niveaux les plus bas.

Niveau 1

Le niveau 1 (ou layer one) définit le lien physique entre deux machines. Il déterminera les signaux à avoir aux différents endroits. Un exemple type est l'Avis V.24 et l'Avis X.21 du CCITT. Les caractéristiques détaillées à ce niveau seront les voltages, les polarités, etc.

Niveau 2

Le niveau 2 définit clairement le lien entre deux équipements, avec les détails de synchronisation, de contrôle de l'intégrité de la transmission, de quittance (acknowledge, ACK) ou rejet (negative acknowledge, NAK) et les retransmissions en découlant. Les protocoles utilisés sont en général les protocoles HDLC, SDLC et BISYNC, ce dernier avec deux variantes : le mode transparent et le mode normal. Le mode transparent permet le passage de n'importe quelle configuration de bits dans un byte, les caractères propres au contrôle de transmission étant reconnus par le fait qu'ils sont précédés par un DLE (data link escape). A noter encore, dans les particularités, le choix du caractère de synchronisation qui est différent en EBCDIC (code des caractères utilisés par IBM) et en ASCII.

Niveau 3

Le niveau 3 est le niveau du paquet de données circulant sur un circuit virtuel établi de bout en bout. Un paquet est un train de données, composé d'au moins un byte, et pouvant en avoir jusqu'à 128, 256, 512 ou 1024 selon la configuration de réseau admise. Ces valeurs ne sont pas exhaustives, mais sont celles configurées en général. Un circuit virtuel est un canal de transmission sur lequel on peut envoyer, sous forme de paquets, des informations d'un bout à l'autre. Un circuit virtuel est aussi parfois appelé circuit de bout en bout. Nous reviendrons plus en détail sur ces notions de circuit en parlant des réseaux publics.

Niveau 4

Le niveau quatre est le niveau de transport, c'est-à-dire le niveau où se fait la mise en paquet et le dépaquetage. Dans les réseaux publics, le programme qui fait ce travail est appelé l'ADP, ou assembleur désassembleur de paquet ou PAD pour packet assembler disassembler.

Niveau 5

Le niveau cinq est le niveau qui permet de faire les appels, qui contrôle les transmissions et les termine. C'est le niveau session.

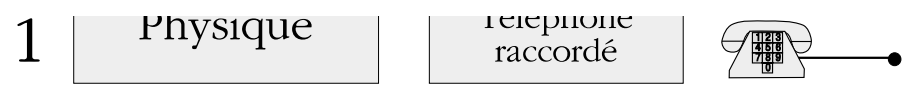
Niveau 6

Le niveau 6 n'est pas nécessairement présent. Sa fonction est la compression des textes pour diminuer les temps de transmission et leurs coûts, éventuellement le niveau de codage/décodage dans des contextes où la confidentialité est importante.

Niveau 7

Le niveau 7 est le niveau de l'application. C'est à ce niveau que les données sont générées ou utilisées. Prenons deux exemples pour expliciter ces niveaux de communication, tous deux tirés de publications spécialisées : Le premier compare le modèle OSI à l'envoi d'une lettre par un courrier postal :

Le second, également tiré des multiples moyens de communications offerts par les PTT, compare le modèle OSI à une communication téléphonique :



Exemple des 7 niveaux du modèle ISO-OSI

Notion de circuit virtuel

Un circuit virtuel est un canal de transmission entre une machine A et une machine B, non nécessairement reliées directement entre elles. Un circuit virtuel est ouvert par la demande de connexion faite par la machine désireuse d'établir une communication. Elle le fait au moyen d'un paquet d'appel qui est reçu par la machine destinataires sous forme d'une notification d'appel. Si l'appel est accepté, un retour se fait par l'émission d'un paquet d'acceptation d'appel qui est reçu par la machine qui a fait la demande sous forme d'un message de connexion établie.

Le circuit est alors opérationnel. Il peut y avoir envoi de paquets de données dans les deux sens.

Pour terminer la communication, un des deux correspondants devra émettre un paquet de libération de circuit qui sera indiqué à son partenaire. Ce dernier doit confirmer la libération du circuit.

Un tel circuit virtuel est appelé un circuit virtuel commuté (switched virtual circuit).

Circuit virtuel permanent

Un circuit virtuel permanent (permanent virtual circuit ou PVC) est un circuit établi de manière fixe et faisant donc partie de la configuration de réseau. Il n'y aura donc ni appel, ni libération de circuit. Ce genre de circuit est utilisé pour connecter des imprimantes par exemple, ou éventuellement pour des raisons de coûts de communication.

Norme X.25 du CCITT

La norme X.25 du CCITT définit de manière très précise, mais en laissant de multiples variantes possibles, les niveaux un, deux et trois du modèle ISO-OSI. Les niveaux quatre et suivants ne sont pas définis dans cette norme et seront en général propres à chaque implémentation.

Contrôle de flux aux niveaux 2 et 3

Un lien de niveau 2 entre deux équipements ou un circuit virtuel au niveau 3 doit pouvoir être contrôlé tout au long de son activité quant à la qualité des transmissions d'une part, au débit d'autre part. Chaque message est quittancé par un "acknowledge" simple ou multiple ou par un "negative ackowledge". Dans ce dernier cas, il y aura retransmission.

Le débit est contrôlé par des messages de contrôle de flux qui peuvent être confondus avec les quittances.

Le nombre de messages non quittancés, tant au niveau deux qu'au niveau trois, est limité à la taille de la fenêtre. Pour ce contrôle, les messages sont numérotés sur trois bits. Mais voyons plutôt un exemple :

Dans cet exemple, la fenêtre est définie à 2, ce qui signifie que la machine émettrice peut envoyer au maximum deux messages non quittancés. Les tailles de fenêtres standard sont de sept au niveau deux et de deux au niveau trois.

Autres Avis de la série X

Quelques autres avis de la série X peuvent nous intéresser ici : les Avis X.3, X.21, X.21.bis, X.28, X.29 et X.75. Nous les citerons simplement avec une brève description de leur contenu, mais sans entrer dans le détail.

Avis X.3

L'avis X.3 définit l'ADP (assembleur désassembleur de paquets, ou PAD, packet assembler disassembler) pour l'accès par un terminal en mode caractère (asynchrone) à un réseau public. L'ADP est en général le programme qui répond aux appels par le réseau téléphonique commuté.

Avis X.21

L'avis X.21 définit l'interface physique entre un ETD (équipement de transmission de données, ordinateur) et un central de réseau. L'avis X.21.bis détermine comment utiliser les modems décrits dans la série d'Avis V sur les réseaux publics.

Avis X.28

L'Avis X.28 détermine comment un terminal travaillant en mode caractère (asynchrone) peut donner des commandes à l'ADP. Les commandes typiques seront les ordres d'appel (établissement d'un circuit virtuel commuté), les modifications des paramètres standard de l'ADP, l'authentification du correspondant (par mot de passe), et les demandes de libération de circuit.

Il s'agit donc d'un microsystème d'exploitation permettant de gérer les ressources du réseau.

Avis X.29

L'avis X.29 définit les mêmes contrôles que l'Avis X.28, mais par des commandes venant de l'ordinateur distant à travers le circuit virtuel. L'entête du paquet de niveau trois permet de définir trois types de messages : les paquets de données (type 0), les paquets de type 1 (contrôle de l'ADP) et les paquets d'interruption (prioritaires). Ce sont les paquets de type 1 qui sont décrits dans cet avis, avec les valeurs par défaut.

Avis X.32

L'avis X.32 permet de transmettre des données entre le réseau X.25 et le réseau téléphonique commuté. En général, on considère deux fonctions : Dial-in et Dial-out. En entrée sur le réseau X.25, il suffit d'avoir un NUI (network user identifier) qui permet de gérer le problème de la taxation et du numéro d'appelant. En sortie, il faut donner une adresse selon le format international E.164 : 094122+++++ pour un numéro à Genève par exemple.

Avis X.75

L'Avis X.75 ne concerne pas directement les utilisateurs de réseaux publics. Il définit les passerelles entre réseaux nationaux.

Une passerelle (gateway) est un lien physique et souvent aussi logiciel entre deux réseaux distincts. Elle doit transmettre les paquets qu'elle reçoit à l'autre réseau en faisant les conversions de protocole nécessaires.

Réseaux existants et matériels connectables

Les réseaux existants sont nombreux. Les premiers à avoir vu le jour sont les réseaux Arpanet et Telenet aux Etats-Unis. Actuellement, la plupart des pays occidentaux ont un réseau public de transmission de données, connu sous un nom ou l'autre. Les constructeurs, quant à eux, ont leur propre réseau, avec parfois la possibilité de passer par un réseau public entre deux noeuds.

Les réseaux publics des différents pays portent des noms tout aussi variés :

Les plus anciens réseaux parmi ceux-ci sont Transpac et PSS, tous deux mis en service en 1981. Le réseau Euronet devrait disparaître au profit de passerelles entre les réseaux nationaux.

Taxation des circuits virtuels

La taxation des transmissions passant à travers un réseau public est basée sur le principe de l'indépendance par rapport à la distance d'une part, sur le volume transmis d'autre part.

Raccordement

Le coût du raccordement sera facturé par mois et dépendra de plusieurs facteurs, facturé par mois :

- vitesse de transmission. Cette dernière est en effet importante quant aux possibilités de raccordement sur un noeud du réseau public. Cette vitesse peut aller pour un raccordement en mode paquet (X.25) de 2400 bauds à 48 kilobauds en Suisse, et de 300 à 1200 bauds en mode caractère (X.3);
- nombre maximum de circuits virtuels pouvant être ouverts à un moment donné; ce nombre permet d'évaluer le nombre de buffers à prévoir au niveau du central du réseau;
- fenêtre maximum au niveau trois; ce paramètre joue également un rôle au niveau du calcul du nombre de buffers nécessaires;
- fenêtre au niveau deux;
- taille maximum des paquets;
- attribution à des groupes fermés d'usagers; (Un groupe fermé d'abonnés (closed user group) est un groupe de raccordements à un réseau public qui permet de limiter les possibilités d'établissement de circuits virtuels. Les possibilités sont de bloquer, vis-à-vis d'autres groupes, les appels sortant, les appels entrant ou tout appel);
- circuits virtuels permanents configurés.

Tous ces paramètres vont permettre de déterminer une taxe mensuelle fixe. Il faudra y ajouter les taxes de communication.

Taxes de communication

L'utilisation du réseau est taxée au volume de transmission principalement, accessoirement au temps de connexion (d'établissement d'un circuit virtuel). Les indications ci-dessous sont les bases de taxation en Suisse sur le réseau Telepac. Toutefois, le principe est analogue dans les autres pays. Les PTT prennent en considération, pour une communication nationale, les paramètres suivants :

- le temps d'établissement d'un circuit virtuel, mesuré entre le paquet d'appel et le paquet de libération. Ce temps n'entre en ligne de compte que pour les circuits virtuels commutés. Il est actuellement de 10 centimes par 10 minutes de connexion. Si un raccordement est utilisé pour plusieurs circuits virtuels, il est évident que chaque circuit virtuel commuté est taxé sur cette base,
- une taxe d'appel de 10 centimes pour chaque paquet d'appel émis,

- le nombre de segments de 64 bytes transmis. Est considéré comme un segment de 64 bytes un groupe de 64 bytes ou moins. Chaque paquet, même vide, transmis est considéré comme étant un segment au moins. Le coût est de 0.125 centime par segment aux heures "plein tarif" et 0.075 centimes aux heures du "tarif réduit" actuellement.

On voit donc que les paramètres significatifs pour un utilisateur à faible trafic seront les taxes mensuelles et, dès qu'un trafic plus soutenu existe, les taxes au volume de transmission. Il ne semble pas être dans les projets actuels des PTT de supprimer les lignes louées et de les remplacer par les réseaux publics de transmission de données. Il existe, en effet, beaucoup de réseaux privés utilisant des lignes louées qui ont des trafics importants et qui nécessiteraient une adaptation importante du hardware du réseau et des taxes. A titre d'exemple, nous avons mesuré expérimentalement le nombre de paquets qu'envoie un élève moyen travaillant en éditeur sur le système time-sharing du CCEES (système Prime de l'enseignement secondaire genevois dans les années 1980-1991) et avons calculé les taxes qui auraient été appliquées s'il y avait passage par un réseau public : environ huit francs pour une session de 30 minutes.

14.- Le RNIS

Depuis l'invention du téléphone en 1876 par Graham Bell, les ingénieurs qui président aux destinées des réseaux de télécommunications à travers le monde, comme les PTT français, Britisch Telecom anglais, Bundespost allemand, ATT américain etc. ont mis en place le réseau de téléphone que nous connaissons actuellement. De n'importe quel point du monde, on peut communiquer avec n'importe quel autre point.

Les réseaux deviennent, avec l'avènement des réseaux "annexes spécialisés", de plus en plus complexe et lourds à gérer : avoir des réseaux différents pour les services différents relève de l'aberration tant économique que de gestion; pourquoi séparer le télex du fax, la transmission de données du téléphone ?

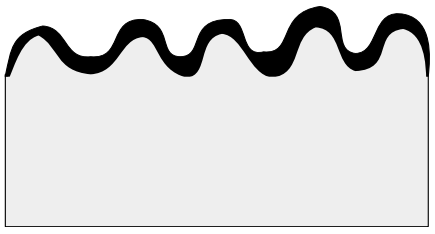
C'est la raison pour laquelle les réseaux téléphoniques du futur ramènent tout à des données numériques, et utilisent ainsi quelque chose d'homogène : le RNIS, ou réseau numérique à intégration de services. Cela va dans le sens d'une uniformisation avantageuse pour l'usager, et d'un meilleur usage des ressource pour l'exploitant du réseau, donc d'une diminution, à moyen terme, des frais.

Le RNIS est un objectif à long terme. Quand on sait que le calcul d'amortissement, pour un central de quartier, se fait sur environ 30 ans, pour les câbles sur à peu près 90 ans, on comprend la durée nécessaire pour cette mise en place. Ce projet devra (presque) tout englober en matière de télécommunications.

En Suisse, ce réseau a pris le nom de Swissnet, 1 pour la transmission de données pures (période 1988 - 1992), puis Swissnet 2 depuis l'intégration de la phonie, donc des passerelles vers le bon vieux téléphone analogique.

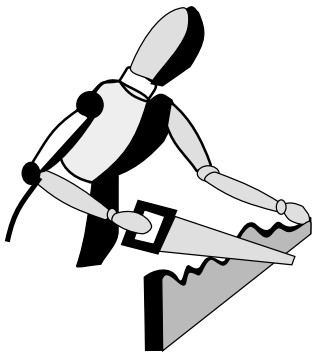
Le principe du RNIS

Pour numériser un signal, il faut d'abord prendre le signal



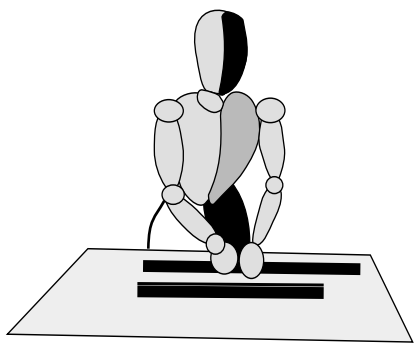
Prendre un signal analogique

qu'il faudra ensuite découper en fines tranches



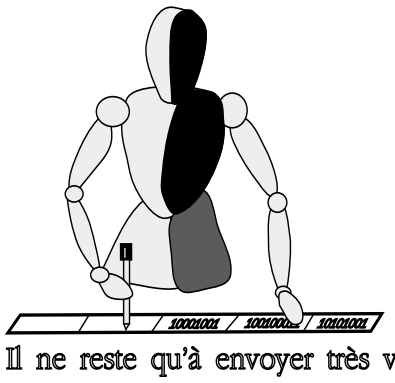
D’abord, découper le signal en fines tranches

que l'on mesurera chacune séparément afin de les envoyer, à la queue-leu-leu, au destinataire



Il faut mesurer la taille de chaque élément

et cela à grande vitesse. A noter qu'il est important de garder le rythme de transfert, car, pour la parole, il n'est pas possible de couper par moment, sinon le son serait haché.



Il ne reste qu’à envoyer très vite...

Mise en place, historique.

L'enjeu était à la fois technique, économique et politique. Les Etats-Unis, le Japon et certains pays européens se sont donc livré à une superbe compétition.

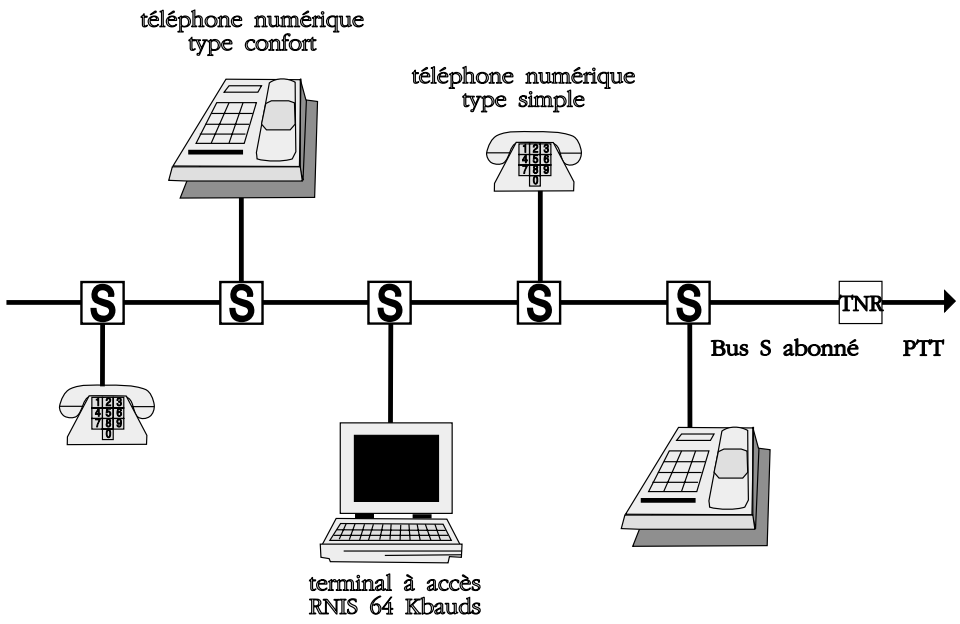
Dans un premier temps, on a procédé à des essais de RNIS sur le terrain et lancé des projets pilotes. Les japonais se sont lancés dès 1984 dans la course au RNIS en montant un réseau INS en suivant les normes du CCITT, avec un essai à Mitaka (près de Tokyo, 1500 terminaux). Quant aux américains, c'est BOC (Bell Operating Companies, sociétés exploitant les réseaux régionaux de télécommunication) qui ont lancé plusieurs opérations, qui étaient opérationnelles vers 1988.

Les européens se sont lancés également dans la course par une directive de la communauté européenne qui date de juin 1986, applicable dès 1987, en vue d'instaurer la reconnaissance mutuelle unique pour tous les pays des procédures de certification concernant le matériel de téléphonie et de transmission de données. C'était un pas important vers l'espace européen commun des télécommunications. La France et l'Allemagne ont été les premiers à se lancer dans l'aventure, suivies par l'Italie, l'Espagne et la Belgique. L'interconnexion devrait, à ce jour, être faite.

Le raccordement de l'usager

Le raccordement de l'usager devait se faire sur le cuivre existant : c'est pourquoi le raccordement de base a des vitesses limitées : 2B+D, ou mieux dit deux canaux B de capacité de 64 kilobits par seconde, et d'un canal D de 16 kilobits par seconde qui sert au contrôle du raccordement, aux messages de service et éventuellement à de la transmission de données par paquets (prévu seulement, en Suisse, pour Swissnet 3). La vitesse de transmission à assurer sur le cuivre est donc de 144 kilobits par seconde. A titre de comparaison, le vidéotex suisse, avec les appareils PTT, ou le minitel français, travaillent à 1200 bits par seconde (1.2 kilobits par seconde !!!).

L'installation domestique se compose donc par une équipement de terminaison de réseau (parfois appelé TNR) et d'un bus S à quatre fils, qui permet le branchement de 8 terminaux quelconques. Ce bus a une longueur maximum de 150 mètres. Chacun des terminaux peut utiliser l'un ou l'autre des deux canaux B, voire les deux. Les types de terminaux connectables directement sont les téléphones RNIS, un terminal informatique RNIS, les fax du groupe 4, des visiophones RNIS ou des PC RNIS. Pour brancher des anciens appareils, on utilisera des adaptateurs de terminaux (appelés TA) qui permettent de recréer un liaison analogiques pour y brancher un bon vieux fax du groupe 3, un modem selon les normes V22 ou V32 par exemple, voire un bon vieux téléphone analogique.



Exemple de configuration - Accès de base

A noter que les PTT suisses accordent, par raccordement de base, dix numéros de téléphone, ce qui permet de choisir pour différents terminaux des adresses différents. Le téléphone RNIS permet de prendre notamment deux adresses, ce qui permet d'avoir un numéro général et un numéro personnel pour chaque membre de la famille par exemple.

Pour les entreprise, il est prévu de disposer d'un raccordement primaire qui, par la fibre ou du coaxial, amène 30 canaux B à 64 kilobits par seconde, plus un canal D à 64 kilobits également, pour la signalisation. Dans ce cas, il est indispensable de disposer d'un autocommutateur privé qui reprend une partie des fonctions du central de quartier.

Service supplémentaires

Les services supplémentaires que les PTT suisses offrent, dans le cadre de Swissnet 2, sont principalement :

- Affichage du numéro de l'appelant : vous savez, avant même d'avoir décroché, qui vous téléphone. Très utile aussi pour des systèmes informatiques qui doivent identifier l'appelant.
- Appel en instance : même si vos deux canaux B sont occupés, on vous signale qu'il y a un appel supplémentaire et on vous fourni le numéro appelant.
- Appel prédéterminé (hot line) : permet l'appel automatique d'un correspondant déterminé.
- Blocage des communications sortantes, avec filtres intercontinental, national ou complet (mais à l'exception des numéros d'urgence).

- Changement de terminal : vous pouvez vous déplacer en "parquant" une communication avant de la reprendre sur un autre terminal.
- Déviation des appels : est possible et simple, de plus inclus dans l'abonnement.
- Numéro d'appel multiple : chaque raccordement de base dispose de dix numéros d'appels. Pour les autocommutateurs, c'est la sélection directe qui est possible.
- Sous-adressage, utile pour les réseaux locaux raccordés.
- Groupe fermé d'usagers, pour limiter les accès soit entrants, soit sortants à une communauté.

Enfin, les coûts restent comparables à ceux du téléphone analogique : aujourd'hui, le raccordement RNIS, qui correspond à deux lignes analogiques, revient au prix de deux lignes. Seule différence : l'usager doit acquérir ses terminaux, qui sont encore nettement plus cher que les téléphones analogiques. Quant aux taxes de communication, elles sont identiques à celles du téléphone analogique, à l'exception d'un appel RNIS-RNIS non abouti qui est taxé 10 centimes pour la fourniture de l'indication du numéro.

Quant à la disponibilité sur le secteur de Genève, elle est bonne : depuis juin 1993, date du raccordement du central de Crassier, seul reste celui de Puplinge qui n'est pas encore câblé avec la fibre : il faudra attendre juin 1994. Les raccordements sont donc disponibles sur tout le canton ou presque, avec des délais d'installation évalués à deux mois.

15.- Les réseaux locaux

Les réseaux locaux peuvent être définis simplement en disant qu'ils se composent de machines diverses connectées entre elles et se trouvant dans le même bâtiment ou groupe de bâtiments. Les distances couvertes sont en général inférieures au kilomètre.

Ces machines sont des ordinateurs, des terminaux, des imprimantes, etc. Dans un contexte plus large, on peut inclure au réseau local d'autres services tels que la télécopie, le télex, le téléphone, la transmission d'image, etc. Dans ce cas, on utilisera un réseau à large bande (broadband) qui permet la transmission simultanée sur un même support (câble ou fibre optique) des services totalement différents.

Dans ce chapitre, nous ne parlerons que de réseaux locaux simples, travaillant en général en bande de base (baseband). A noter que l'installation d'un réseau à large bande est souvent beaucoup plus onéreuse, car le câble doit être de caractéristiques bien précises et même les coudes faits dans le câble peuvent influencer la qualité de la transmission.

Topologie des réseaux

Les réseaux locaux peuvent être de différentes topologies, ayant toutes leurs avantages et leurs inconvénients. Dans les paragraphes qui suivent, nous allons décrire les principales, en détaillant les plus intéressantes.

Liaison point à point

Ce type de réseau n'est cité ici que pour mémoire, car nous ne savons pas si on peut vraiment parler de réseau. Il s'agit en fait de la simple connexion de deux machines entre elles, ou d'un terminal à un ordinateur. Le câblage dépendra principalement des caractéristiques respectives des matériels connectés.

Réseau en étoile

Le réseau en étoile est le réseau type des années 60 et 70. Chaque terminal est connecté sur le site central par son propre câblage.

L'avantage de ce type de réseau est le fait de ne pas avoir de saturation au niveau des lignes de transmission, puisque chacune d'elles peut travailler à la vitesse nominale pour un seul utilisateur. Aucun protocole particulier n'est donc nécessaire.

Un inconvénient de taille est par contre le câblage important nécessaire : il faut, en effet, autant de câbles (voire de modems) qui arrivent sur le site central. De plus, une panne quelconque de cet élément central entraîne une panne générale de tout le réseau.

Nous parlerons également de réseau en étoile même lorsque les lignes de plusieurs terminaux distants, mais situés dans le même bâtiment, sont multiplexées par une seule ligne.

Réseau en anneau

Le réseau en anneau est un cas particulier, car il décentralise le contrôle tout en ne garantissant pas, dans sa version simple, une fiabilité plus grande que le réseau en étoile. Mais nous verrons, après avoir expliqué le principe, les améliorations qu'il peut avoir pour garantir une bonne fiabilité.

Le principe consiste à avoir une boucle de transmission sur laquelle tous les postes sont connectés :

Lorsque l'ordinateur A, par exemple, dispose du jeton (ou token), il met un message sur l'anneau. Ce message passe par B, C et D pour revenir sur A. L'ordinateur destinataire copie le message en même temps qu'il le passe au suivant. Si le destinataire s'est reconnu dans l'adresse qui se trouve en tête du paquet, il l'indique en modifiant le byte de contrôle pour indiquer "message copié". Chaque noeud retarde le message du temps de transmission d'un bit. Après avoir envoyé son message, le jeton doit être passé au suivant, et ainsi de suite.

L'avantage de ce type de réseau tient au fait que le temps d'attente maximum avant de pouvoir transmettre un message est connu (nombre de noeuds fois la durée de transmission d'un message). Il n'y aura pas de saturation pouvant engendrer des erreurs de transmission et le câblage est moins important que celui d'un réseau en étoile s'il a été bien conçu.

L'inconvénient majeur est le fait qu'une interruption à un endroit quelconque de l'anneau empêche toute transmission sur tout le réseau. C'est pourquoi une version légèrement différente de ce type de réseau est utilisée :

Le point de liaison de chaque ordinateur sur l'anneau se fait dans un boîte de jonction, qui contiendra, par exemple, un simple relais qui, en cas d'absence de contrôle ou de dérangement sur l'ordinateur, ferme la boucle. Cette manière de faire perturbera le réseau au maximum lors du changement d'état du relais, et laisse ainsi l'anneau en état de fonctionnement.

Les interfaces hardware que l'on trouve pour les micro-ordinateurs actuellement sur le marché ont un protocole HDLC avec l'option du jeton et une capacité d'adressage de 255 stations sur un anneau.

Réseau multi-étoilé

Comme type de réseau plus complexe, on trouve les réseaux multi-étoilés. Ce type de réseau a les mêmes avantages et inconvénients que le réseau en étoile, avec toutefois un point positif au niveau du câblage : un seul lien suffit entre le noeud central et le noeud qui forme l'étoile distante.

De plus, avec un réseau multi-étoilé, il est possible de d'augmenter la complexité en mettant plusieurs étoiles en profondeur. Le réseau devient alors hiérarchisé.

Réseau en bus

Le réseau en bus est certainement celui qui est le plus simple à câbler. Il suffit, en effet, d'un câble qui fasse le tour de tous les postes de travail. Les caractéristiques de ce câble dépendent des interfaces : cela va du simple câble à deux fils torsadés au câble coaxial. Les facteurs influençant le choix sont la vitesse et le type de modulation utilisés.

Lorsqu'un poste veut émettre sur le réseau, il doit d'abord écouter la ligne pour voir si elle est libre et, dans ce cas, il peut émettre. Dans le cas où deux postes commencent à émettre en même temps, ils vont le détecter, car ils contrôlent toujours la ligne afin de s'assurer de ne pas être brouillés.

Au cas où une collision devrait avoir lieu, celui qui la détecte brouille l'autre correspondant, puis tire un nombre aléatoire et attend un temps correspondant au nombre tiré.

Cette technique est connue sous le nom de CSMA/CD (carrier sense multiple acces/collision detection), et également sous le nom d'ethernet.

En fin de ligne, des deux côtés, on installe un "terminator", c'est-à-dire une vulgaire résistance qui a pour but d'absorber l'information et ainsi d'éviter un écho des données.

La vitesse de transmission d'un réseau ethernet est de 10 mégabits. Toutefois il existe des versions dérivées travaillant entre 1 et 100 mégabits.

L'Ethernet

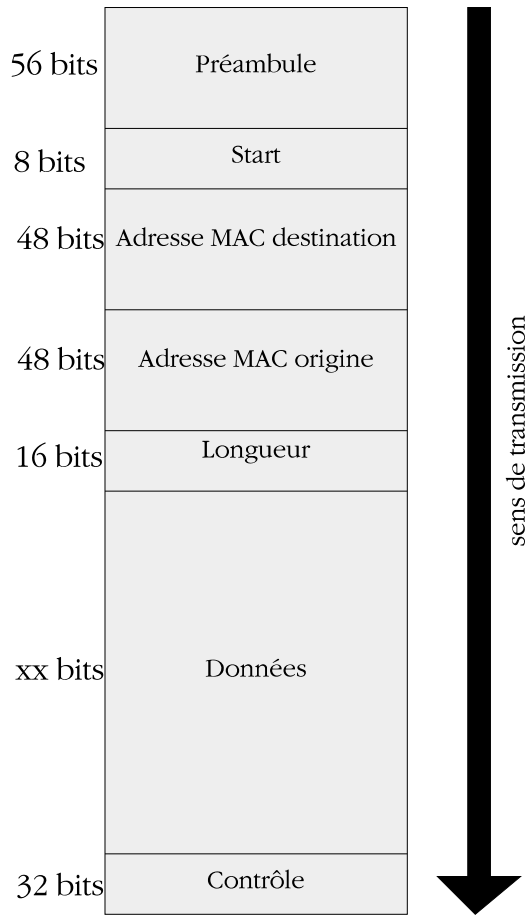
L'éthernet est le fruit de la collaboration entre Xerox, Intel et DEC à la fin des années 70. Dès 1972, Xerox se mit à la réalisation d'un réseau local à haut débit (1 à 10 Mbits par seconde), permettant un grand nombre de station reliées, avec des droits d'accès égaux des postes face au réseau, et sans composant central. Les premiers résultats probants datent de 1976, le premier standard de 1980.

L'éthernet a été normalisé par l'IEEE (Institute of Electrical and Electronical Engineers sous le nome IEE 802.3, également adoptée par l'ISO comme ISO 8802.3.

En réalité la norme ISO 8802.3 se décompose en trois normes :

- ISO 8802.3 10 base 5 (éthernet épais)
- ISO 8802.3 10 base 2 (cheapernet)
- ISO 8802.3 10 broad 36 (peu utilisé)
- ISO 8802.3 1 base 5 (obsolète)
- ISO 8802.3 10 base T (starlan, torsadé)

L'éthernet utilise une transmission par trame ou paquets, qui comportent un préambule de 56 bits (max 64), suivi d'un octet start indiquant le début, puis 6 octets d'adresse origine, 6 octets d'adresse destination, la longueur des données sur 2 octets, les données elles-même et enfin une CRC servant à valider la transmission.



Ethernet Epais (ISO 8802.3 10 base 5)

La norme ISO 8802.3 10 base 5, aussi appelé "thick ethernet", utilise un protocole d'accès CSMA/CD à 10 Mbits par seconde en bande de base sur un câble coaxial à double blindage, de couleur jaune. Il est terminé par une résistance de 50 Ohms à chaque bout. La taille maximum d'un segment est de 500 mètres, en ne traversant que 4 répéteurs au maximum.

La limite des 4 répéteurs est en fait liée à deux contraintes : la première est le temps de réaction du répéteur, qui génère un retard de l'ordre de 1 à 2 bits. L'entête d'une trame éthernet est de 56 bits, qui peut prendre au maximum 64 bits. Environ 4 répéteurs nous amènent donc à cette limite. La seconde est liée au temps de propagation du signal et à la détection de collision. Le délai de réémission d'une trame étant un multiple de 51,2 microsecondes, on a ramené le délai de détection de collision à 45 ms par sécurité, soit 22.5 microseconde dans un sens et 22.5 dans l'autre.

Le nombre de postes maximum branché sur un segment de câble épais est de 100. Comme on peut brancher un poste qu'à des emplacements fixes distants de 2.5 mètres l'un de l'autre, on pourrait doubler ce nombre. Mais ce sont des caractéristiques électriques qui imposent la limite de 100 postes.

Chaque poste est raccordé au câble par un transceiver qui est, en général, serti sur le câble jaune. Il est raccordé au poste par un câble AUI (Attachement Unit Interface) à la carte réseau de l'ordinateur. Le câble AUI peut avoir une longueur maximum de 50 mètres, a 4 paires torsadées et peut avoir jusqu'à 20 mm de diamètre selon la longueur. La prise est de type D-sub à 15 pôles, ce qui nécessite des trous de tirage de câble de 40 mm.

En résumé, voici les caractéristiques de l'éthernet épais :

- norme ISO 8802.3 10 base 5
- vitesse de 10 Mbits/seconde
- transmission en bande de base
- câble jaune à double blindage
- rayon de courbure d'au moins 25 cm
- mise à terre du blindage en un seul point
- résistance de 50 Ohms 1 Watt à chaque bout
- pas de dérivation sur le câble sans répéteur
- longueur minimale d'un segment de 23.4 mètres
- longueur maximale d'un segment de 500 mètres par segment
- rayon de courbure de 20 cm.
- maximum de 100 postes sur un segment
- distance entre deux postes d'un multiple de 2.5 mètres
- maximum de 4 répéteurs entre deux postes quelconques du réseau.

Ethernet fin (ISO 8802.3 10 base 2)

L'Ethernet fin, connu aussi sous le nom de cheapernet ou éthernet fin, permet une installation meilleur marché et plus facile à adapter aux contraintes architecturales, le rayon de courbure et le diamètre étant plus petits.

Il permet au maximum 30 machines par segment, n'a qu'un seul blindage et est limité en distance à 185 mètres. Les postes ou les transceiver sont raccordés par des prises BNC. Au cas où on utilise le transceiver de la carte réseau, il convient d'amener le câble réseau jusqu'à la carte, d'y brancher un T et de repartir de la carte. La dérivation du câble ne doit par dépasser une longueur de zéro ! Il est par contre possible de mettre un transceiver, auquel cas la distance entre le transceiver et la carte réseau peut monter, comme pour l'éthernet épais, à 50 mètres au moyen d'un câble AUI.

Une autre solution consiste à utiliser des "tap" qui sont des boîtiers de connexion avec prise ininterruptible. La boîte sans câble compte alors pour 1.20 mètres, chaque boîtier avec câble pour 7 mètres + 2 fois la longueur du câble. On voit que cette solution coûte cher en mètres et réduit sensiblement la longueur d'un segment.

En résumé, voici les caractéristiques de l'éthernet fin :

- norme ISO 8802.3 10 base 2
- vitesse de 10 Mbits/seconde
- transmission en bande de base
- câble coaxial à simple blindage type RG 58 C/U
- prises BNC 50 Ohms pour les raccordements
- mise à terre du blindage en un seul point
- résistance de 50 Ohms 1 Watt à chaque bout
- pas de dérivation sur le câble sans répéteur
- longueur maximale d'un segment de 185 mètres par segment
- maximum de 30 postes sur un segment
- distance entre deux postes de 0.5 mètres au moins
- maximum de 4 répéteurs entre deux postes quelconques du réseau.

Ethernet torsadé (ISO 8802.3 10 base T)

La norme ISO 8802.3 10 base T découle de la norme ISO 8802.3 1 base 5. Elle utilise une topologie de câblage de type "starlan" (multiétoilé). Le nombre maximum de postes n'est pas défini. La longueur maximum d'un segment est de 100 mètres. La connexion entre les segments se fait au moyen de "hubs" qui jouent le rôle de répéteurs.

Le câble est en paire torsadée non blindée de type 1 ou 2 (IBM) ou G87. En principe, on pourrait utiliser l'infrastructure téléphonique existante. La transmission se fait également en bande de base.

Autres support pour l'éthernet

Un autre support de transmission qui permet des distances plus longue est l'utilisation de moyens optiques, soit par de la fibre, soit par laser. L'avantage de ces deux méthodes réside en des distances plus grandes possibles, mais sans connexion intermédiaire, rendant le piratage plus difficile.

La distance que permettent ces moyens va de l'ordre du kilomètre (laser, fibre multimode) à plusieurs kilomètres (fibre monomode). Le lien se fait toujours par des câbles AUI entre le répéteur, bridge, routeur ou carte réseau et le transceiver.

Passerelles

Pour passer d'un réseau local à un autre, qu'il soit de même type ou non, on utilisera des passerelles (gateway). Il s'agit d'un ordinateur (mini ou micro) qui est relié aux deux réseaux entre lesquels il doit faire la jonction, éventuellement par l'intermédiaire d'un modem.

Sa fonction première est de convertir les messages de l'un des réseaux dans le format de l'autre, et vice-versa, d'où le nom parfois donné de convertisseurs de protocoles ou de boîte noire (black box).

D'autre part, afin de ne pas saturer le réseau distant et par là ne pas augmenter inconsidérément les coûts de transmission, la passerelle doit opérer un tri et ne passer que le strict nécessaire dans les meilleures conditions au réseau distant.

Il arrive parfois qu'à ces fonctions soit encore associée une fonction de facturation par le comptage des paquets et de la durée de connexion.

A titre d'exemple, les passerelles entre les réseaux publics nationaux doivent répondre aux normes décrites dans l'Avis X.75 du CCITT et les passerelles le plus fréquemment disponibles sur le marché sont des passerelles permettant l'accès aux réseaux publics en X.25 ou au réseau SNA ou 3270 d'IBM.

Implémentation d'un réseau local dans un OS

Le problème de l'implémentation d'un réseau local dans un système d'exploitation pose toujours le problème lancinant de la transparence, c'est-à-dire de l'assurance que le réseau n'influence pas les logiciels utilisés. Les fonctions d'un réseau local dans un contexte de plusieurs microsystèmes sont principalement de pouvoir partager la périphérie, à savoir les disques durs, les imprimantes, les passerelles avec d'autres réseaux ayant des ressources intéressantes (vidéotex, bases de données, etc.).

Quatre techniques sont principalement utilisées pour une telle implantation :
`.i.BIOS`

- en remplaçant le BIOS (Basic Input-Output System) qui se charge des opérations de base liées au hardware par un BIOS contenant les drivers modifiés pour le réseau,
- en mettant le logiciel de réseau entre le BIOS et le système d'exploitation,

- en remplaçant le système d'exploitation par un autre qui supporte les fonctions de réseau,
- en mettant le logiciel de réseau entre les programmes utilitaires et le noyau du système d'exploitation.

Montrons cela au moyen de quelques schémas et expliquons les avantages et inconvénients de chacune de ces méthodes :

Au niveau du BIOS

Le BIOS étant le logiciel de niveau le plus bas pour gérer la périphérie, il semble tout indiqué de mettre le logiciel de réseau à ce niveau :

Le passage d'une machine se fait au niveau du BIOS. Cela signifie que le logiciel de base doit être strictement identique dans les différentes machines, car on peut aisément deviner ce qui se passerait si une structure de répertoire différente existait sur deux machines traitant chacune le même enregistrement physique à sa manière. De plus, l'accès concurrent à un enregistrement peut poser problème.

Entre l'OS et le BIOS

Une autre technique consiste à interfacer le réseau entre l'OS et le BIOS. L'avantage par rapport à la méthode précédente est l'indépendance du logiciel de réseau du BIOS, ce qui permet éventuellement des révisions différentes du BIOS. A noter que les constructeurs n'assurent en général pas de compatibilité à ce niveau entre révisions.

Les BIOS respectifs ne sont pas modifiés !

Au niveau de l'OS

Si le logiciel de réseau se trouve au niveau du système d'exploitation, on aura une structure de ce type :

L'inconvénient principal est la nécessité de disposer d'un système d'exploitation avec le logiciel de réseau inclus. Il y aura donc parfois divergence entre les niveaux de révision du système d'exploitation avec réseau et celui sans réseau.

Au-dessus du système d'exploitation

Le logiciel de réseau intercalé entre les programmes utilitaires (voire le processeur de commandes) et le système d'exploitation est certainement la meilleure méthode :

Mis à ce niveau, le logiciel intercepte toutes les requêtes au système d'exploitation et les transmet, si besoin est, à la machine distante pour exécuter la requête. Comme la compatibilité ascendante au moins doit être garantie pour respecter le sérieux du fournisseur, cette technique est certainement la meilleure. De plus, elle assure un fonctionnement même avec des révisions du logiciel différentes. Enfin, elle permet de garantir une intégrité du système de gestion des fichiers, car c'est toujours le gestionnaire de fichier local qui traite les requêtes. L'accès concurrent dans un contexte de bases de données reste toutefois un problème qui est relativement simple à résoudre.

16.- Contrôle de l'accès aux ressources par les réseaux

Comme nous l'avons vu, les réseaux publics et les réseaux locaux permettent des échanges d'informations rapides et en grand nombre. Il se pose donc le problème du contrôle de l'accès aux ressources et aux fichiers, d'une part pour des questions de respect de la sphère privée, d'autre part pour ne pas pénaliser les utilisateurs tolérés d'un système ou encore pour pouvoir facturer les ressources utilisées. L'authentification d'un demandeur de ressources ou d'informations est donc particulièrement important sur les réseaux qui ont un lien avec l'extérieur ou pour lesquels des accès en libre service existent.

Une phase d'identification, souvent appelée login ou logon, permet de s'assurer que l'utilisateur est bien autorisé à travailler. Cette autorisation reste valable jusqu'à ce l'utilisateur fasse un logoff ou logout. Comme il arrive que, par oubli, il ne le fasse pas, on implémentera en général un temps d'inactivité au bout duquel la déconnexion sera faite d'office.

Mais voyons quelques méthodes souvent utilisées pour s'assurer de la validité d'un login.

Phase d'identification de l'utilisateur

Technique du mot de passe

La technique la plus souvent utilisée dans un contexte time-sharing est celle du mot de passe. Avant d'autoriser quelque transaction que ce soit, l'utilisateur est invité à décliner son identité et à donner son mot de passe. En cas de correspondance avec les informations stockées dans le système, l'utilisateur sera autorisé, dans les limites définies dans son profil, à utiliser des ressources.

Le problème des mots de passe est toujours important et sujet à discussion. Combien de personnes, souvent non informaticiennes, laissent-elles le mot de passe inscrit sur le terminal, dans une tirette du bureau, ou encore le choisissent en mettant leur prénom ou celui de leur conjoint.

Un mot de passe devrait avoir au moins six caractères que l'utilisateur devra taper en "aveugle", c'est-à-dire sans écho.

Double mot de passe

Pour améliorer la sécurité, une technique parfois utilisée consiste à disposer, pour chaque utilisateur, d'une dizaine (ou plus) de questions pour lesquelles on attend une réponse circonstanciée. Si le premier mot de passe est correct, un générateur de nombres aléatoires permet de tirer une des questions que l'on pose. L'utilisateur doit donner la bonne réponse, si possible en "aveugle", faute de quoi il est rejeté.

Contrôle par signature

Une technique relativement fiable est la signature demandée à l'utilisateur et comparée à celle stockée en machine. Des algorithmes analysant entre autres la vitesse et les accélérations lors du traçage permettent d'authentifier une signature même lorsqu'elle n'est pas tout à fait identique.

Carte magnétique

Une autre possibilité consiste à devoir introduire une carte magnétique dans un appareil situé près du terminal. Si le code enregistré sur la carte magnétique correspond, l'utilisateur est accepté. L'inconvénient majeur de ce système est l'oubli, quasi chronique, de ces cartes sur les terminaux ou leur perte. La meilleure technique est donc d'associer à la carte magnétique un mot de passe qui ne doit pas avoir de redondance avec la carte et que l'utilisateur devra donner comme décrit auparavant.

"Carte à puce"

La technique de la carte à puce offre une meilleure sécurité que la carte à bande magnétique, qui est relativement facile à copier ou à falsifier une fois l'algorithme découvert. Conçu comme un processeur avec un bout de code et des données, il faut l'alimenter et transmettre des données lors de l'utilisation. Le point de contrôle devra donc être équipé spécialement.

"Securid"

Le système "Securid" est une alternative à la carte à puce. La carte, qui contient de l'électronique, fournit des nombres calculés algorithmiquement en fonction du temps. L'authentification du client se fait par transmission du mot de passe que fournit la carte. L'avantage est d'avoir une séquence de nombres et de ne pas devoir disposer, sur le terminal ou le point de contrôle, d'autre équipement qu'un clavier numérique. Le système informatique sur lequel l'identification se fera dispose d'une routine qui utilise le même algorithme.

Les séquences sont programmées pour une certaine durée, en général de l'ordre de trois ans au maximum. A ce jour, on considère ce système comme permettant une bonne sécurité.

Autres techniques

Beaucoup d'autres techniques ont été proposées. Elles nécessitent toute du hardware spécifique. Citons, à titre d'exemple, l'analyse des empreintes digitales, l'examen de la rétine, la reconnaissance de la voix, etc.

Lieu où le contrôle doit s'effectuer

Le lieu où le contrôle des droits d'accès doit se faire est l'endroit où la ressource se trouve, éventuellement au point d'entrée dans le réseau. En aucun cas il ne devra être fait sur une machine distante, ce qui était le cas de terminaux plus ou moins sophistiqués d'un réseau publique de ressources. La technique utilisée, facile à pirater, était d'envoyer, depuis le terminal, le nom d'utilisateur; le système central renvoyait alors le mot de passe au terminal qui le comparait à celui fourni par l'utilisateur. Il est évident que, moyennant un petit programme bien fait, mais simple, il était possible de se faire une librairie de mots de passe.

Limite des techniques de contrôle

Il est bien clair que toutes les techniques, aussi sophistiquées soient-elles, ont des limites. Toutes celles qui nécessitent du matériel spécifique d'identification ne seront valables que sur un réseau fermé, c'est-à-dire un réseau où aucune autre machine ne peut se connecter que celles prévues, avec leur logiciel. On court donc un risque dès que l'on se trouve dans un contexte de programmation sur la machine distante. La seule méthode utilisable reste donc celle d'un mot de passe généré soit par l'utilisateur (cas classique), soit par un dispositif quelconque.

Contrôle dans un réseau commuté

Lorsque les lignes louées ne peuvent être une solution au problème de la sécurité d'un système informatique, que cela soit pour des raisons de coût ou de topologie du réseau, le réseau commuté devient une solution relativement bonne. Elle pose toutefois un problème ardu de sécurité.

Comment, dès lors, garantir un niveau de sécurité suffisant ? Les systèmes de modems permettent actuellement, sans coût supplémentaire prohibitif si le nombre de lignes dépasse la dizaine, de tester si l'utilisateur est autorisé. Les niveaux de sécurité offerts vont de rien (tout appel entrant est transmis sur le port V24 du modem) à un contrôle par nom et mot de passe, voire encore par un rappel sur des numéros préenregistrés, n'autorisant ainsi la connexion que depuis un seul lieu physique.

Quant au fait de valider un numéro par le numéro de téléphone appelant, les PTT suisses ne le fournissent pas encore, ne le pouvant pas toujours, car le réseau suisse n'est pas encore entièrement équipé de centraux à commande numérique. En outre, il n'est pas sûr que cette facilité sera offerte dans le futur, car des problèmes juridique de respect de la sphère privée seront invoqués. (Les administration des téléphones des Etats Unis fournissent cette information, mais risquent de bientôt la supprimer, certains tribunaux ayant créé une jurisprudence qui les obligera à ne plus fournir ces numéros.

Le réseau swissnet (RNIS suisse), dans sa phase 2 qui démarre fin 92, permettra, selon l'état de nos connaissance, d'obtenir le numéro appelant. Comme il s'agira tant de téléphone que de transmission de données, et en l'état de la législation et de sa jurisprudence, il n'est pas possible d'affirmer quelle sera la règle par la suite.

Contrôle de l'origine de l'appel

Dans les réseaux publics de transmission de données travaillant en X.25, il y a moyen de contrôler l'origine de l'appel, puisque le paquet d'appel contient l'adresse de l'abonné appelant. Ce dernier peut le fournir et, s'il le fournit, il doit être correct. Si ce numéro n'est pas donné, c'est le réseau qui le rajoute en fonction de la ligne physique utilisée. Le problème est donc repoussé au niveau du réseau public qui doit être fiable sous cet aspect. Lorsque l'abonné entre sur le réseau public par une ligne téléphonique commutée connectée à un ADP, ce dernier devra valider l'abonné (chose faite au moyen d'un mot de passe...).

Dans le réseau téléphonique commuté, les nouveaux centraux permettent de connaître l'origine d'un appel. En Suisse toutefois, cette possibilité n'existe pas encore pour l'usager, mais elle sera peut-être introduite prochainement. Il s'agira alors de décoder le numéro de téléphone appelant, et de le valider. A noter que ce service est disponible pour les raccordement swissnet 2 (RNIS) pour les appels RNIS-RNIS.

Autres mesures de sécurité

Parmi les autres mesures, nous en citerons quelques-unes. Cette liste n'est donc de loin pas exhaustive.

Traçages

La technique consistant à tracer les transmissions, partiellement ou complètement, permet de savoir si le correspondant utilise des ressources non autorisées. Le principal problème qui y est associé est celui du dépouillement : un traçage a en effet une valeur qui est inversement proportionnelle à son poids.

Traçage des erreurs de validation

Une méthode plus efficace est le traçage des erreurs de validation des utilisateurs (mots de passe erronés) avec indication, si possible, de l'origine de l'appel et des noms utilisés. Un tel traçage peut avoir lieu sur un terminal ad hoc ou dans un fichier. Il n'a de valeur que s'il est contrôlé.

Avertissement d'un opérateur

Les utilisateurs demandant des ressources "sensibles" peuvent devoir demander à un opérateur les droits supplémentaires dont ils ont besoin. Ce dernier devra quittancer la demande en les allouant ou en les refusant. Une autre technique consiste simplement à attirer l'attention de l'opérateur par un message circonstancié.

Comptabilisation

Toute machine disposant de ressources particulières devrait comptabiliser, de manière aussi détaillée que possible, les ressources allouées. Une facturation, même pro format, devrait être faite pour que l'utilisateur, son chef de projet ou toute autre personne puisse se rendre compte que quelqu'un d'autre a utilisé des ressources en son nom. Une technique simple consiste simplement à indiquer la dernière fois qu'elle a utilisé les ressources, avec date et heure.

Invalidation

Il est possible également d'envisager une invalidation des droits d'accès aux ressources si l'utilisateur (ou son pirateur) se trompe plus d'un certain nombre de fois dans un mot de passe. L'utilisateur devra alors voir l'administrateur du système afin que celui-ci rétablisse les droits.

Rupture de la liaison physique

Afin de décourager un pirateur faisant des essais systématiques sur un réseau commuté, il y a moyen de ne laisser qu'un temps restreint pour le processus d'identification et de limiter les essais avec mot de passe erroné à un, deux ou trois, après quoi on libère le circuit commuté ou la ligne téléphonique.

Introduction de délais

Il est également possible de pénaliser l'utilisateur ayant fait un mot de passe erroné ou ayant demandé une ressource à laquelle il n'a pas droit en le faisant attendre pendant un délai de quelques secondes. Le délai devient ainsi prohibitif pour des essais systématiques.

17.- La messagerie

Après l'explosion de l'informatique individuelle, nous serons bientôt confrontés au développement important de l'informatique de groupe. Et, dans ce domaine, la première pierre est la mise en place d'un système de messagerie électronique. Actuellement, un flot important d'informations circule entre les membres d'une communauté administrative, entre services et départements, voire entre les entreprises. Ce flux est essentiellement composé de papier dont le transport reste lent et coûteux. L'objet à atteindre est donc de limiter le transport du papier, à défaut du papier lui-même puisque les gens préfèrent encore lire un mémo qu'il peuvent manipuler, crayonner, graphiter, ...

L'objectif à atteindre avec un système de messagerie est donc d'accélérer le transport de ces informations, que ce soient des notes, des budgets, des rapports, des projets, ...

Le succès d'un système de messagerie dépend du nombre d'abonnés raccordés d'une part, de la facilité d'accès d'autre part. La messagerie doit, en outre, être une facilité ajoutée à un poste de travail, et pas un clavier supplémentaire rajouté sur la table de travail.

Un service de boîte aux lettres électroniques permettra un gain de temps important par rapport au téléphone : pas besoin, en effet, de trouver son correspondant qui n'est jamais disponible... il suffit de glisser dans sa boîte au lettres un petit mémo expliquant ce que l'on désire, et il le lira au moment fixé par lui, indépendamment des disponibilités de l'auteur. Mais des écueils classiques peuvent survenir.

Premièrement, il est indispensable que les partenaires à un tel système ouvrent périodiquement leur boîte aux lettre pour voir si du courrier est arrivé. (Il existe des systèmes qui signalent, lors de chaque connexion au système central, qu'il y a du courrier à lire et, oh raffinement suprême, avertissent même, lors de l'arrivée d'un message, l'utilisateur.

Deuxièmement, il faut que le système permette de savoir si un message peut être délivré à son destinataire ou non (aspect fiabilité) et que l'endroit où sont stockés les messages soit protégé contre des accès non autorisés.

La normalisation en matière de messagerie

Les systèmes de commutation de messages et la messagerie se sont développés jusqu'en 1980 en absence de toute normalisation. Quelques normes disparates existaient que pour des services tels que le télex (norme F.31) ou entre compagnies de transport aérien (normes ATA-IATA sur le réseau SITA (*Société Internationale des Télécommunications Aéronautiques*)) ou encore dans les milieux bancaires (réseau SWIFT (*Society of Worlwide Interbank Financial Telecommunication*)).

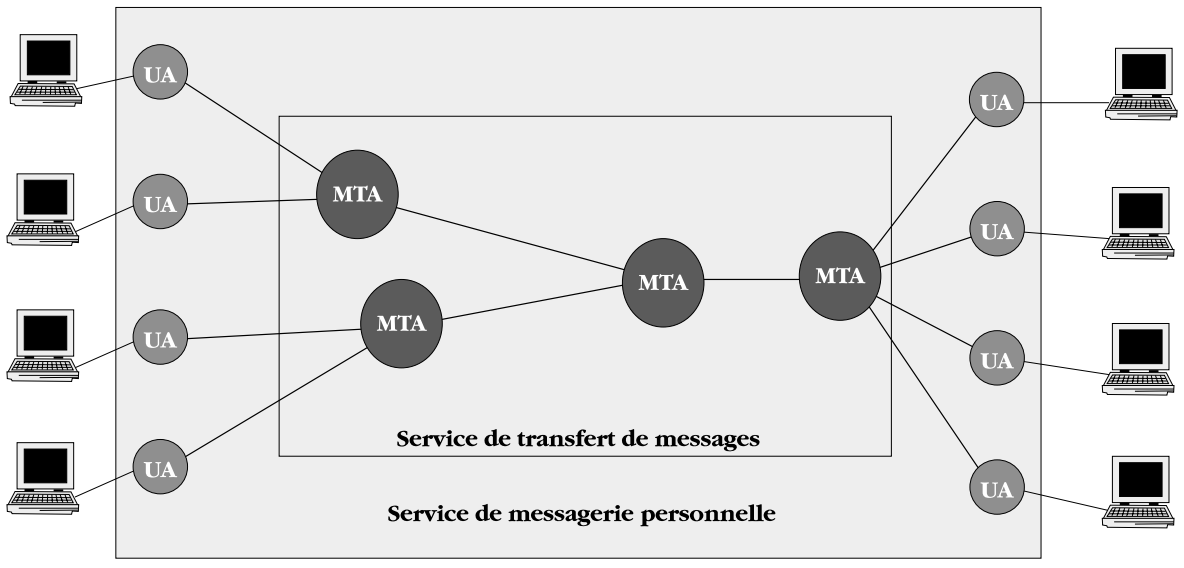
L'IFIP (*International Federation for Information Processing*) ont servi de détonateur et de base au CCITT qui a défini huit normes de la gamme "X-MHS" (MHS pour *Message Handling System*) ou normes X.400. La norme définit un modèle fonctionnel en trois entités :

- l'utilisateur représente l'émetteur effectif ou le récepteur final d'un message. Il peut s'agir d'une personne ou d'un processus informatique utilisant la messagerie.

- l'Agent Utilisateur (UA, *User Agent*) est l'entité avec laquelle l'utilisateur interagit pour préparer des messages, les émettre, les recevoir et il met à disposition quelque autre services en relation avec la messagerie.

- l'Agent de Transfert des Messages (MTA, *Message Transfert Agent*) constitue, avec d'autres MTA, un système de transfert de messages qui assure l'acheminement du message de l'UA émetteur à l'UA destinataire.

L'utilisateur qui désire envoyer un message le prépare donc au moyen de son UA préféré, puis le dépose sur le MTA auquel il est rattaché. Le système de transfert de messages l'achemine à travers un ou plusieurs MTA vers le MTA destinataire. Si la remise peut être effectuée à l'UA destinataire, un avis de remise facultatif peut être renvoyé à l'expéditeur, sinon, c'est un avis de non-remise qui est renvoyé.



L'implantation des UA et MTA peut être faite de différentes manières, la frontière étant très bien marquée chez certains fournisseurs, nettement moins bien chez d'autres.

Les services de messagerie utilisent un protocole selon le modèle ISO-OSI sur les couches 1 à 6.

- La couche messagerie interpersonnelle (IPM, *InterPersonnal Messaging*) décrite dans l'avis X.420 définit les agents utilisateur de la catégorie "courrier électronique" en spécifiant les entêtes des messages, ressemblant étrangement à des notes de service.

- Le service de transfert de messages, décrit dans l'avis X.411, sur lequel est basé la messagerie interpersonnelle, mais qui aussi servir à des transferts de fichiers.

- Le protocole P1, qui définit clairement l'interconnexion entre deux MTA

- Le protocole P3, qui définit la liaison entre le MTA et l'UA, qui est défini dans la norme X.410.

Un message se compose donc d'une enveloppe P1, qui contiendra tout ce qui nécessaire aux système de transfert de messages d'acheminer le message au destinataire. En principe, l'usager n'a pas accès au protocole P1, donc à l'enveloppe P1. Cette enveloppe est suivie d'un entête P2 qui contient en fait l'entête d'une note de service et les indication pour l'UA. Cet entête est enfin suivi par un ou plusieurs corps de message qui peuvent être de types différents.

Les corps de messages peuvent être de type :

IA5 : corps de type texte, structuré en lignes séparées par des retour de chariot et avance de ligne, utilisant un alphabet à 7 bits, c'est-à-dire sans accents. C'est le plus fréquemment utilisé, car implanté dans tous les systèmes de messagerie,

ISO 6937 : Code à 8 bits permettant les accents. Ces derniers sont codés sous forme de deux caractères, l'accent avant la lettre

T61 : presque identique à ISO 6937.

Les autres types cités ci-dessous sont en général appelés "pièces jointes". On les utilisera pour passer des documents réalisé en traitement de texte par exemple, avec toutefois la contrainte que le destinataire devra disposer du même traitement de texte pour pouvoir ouvrir le document.

Voix : pièce de type vocal, non encore normalisée.

Tif0 et Tif1 : pièce de type image bitmap. Plusieurs types existent.

G3fax : pièce de type fax du groupe 3. Le corps se compose d'un ensemble de page passée graphiquement.

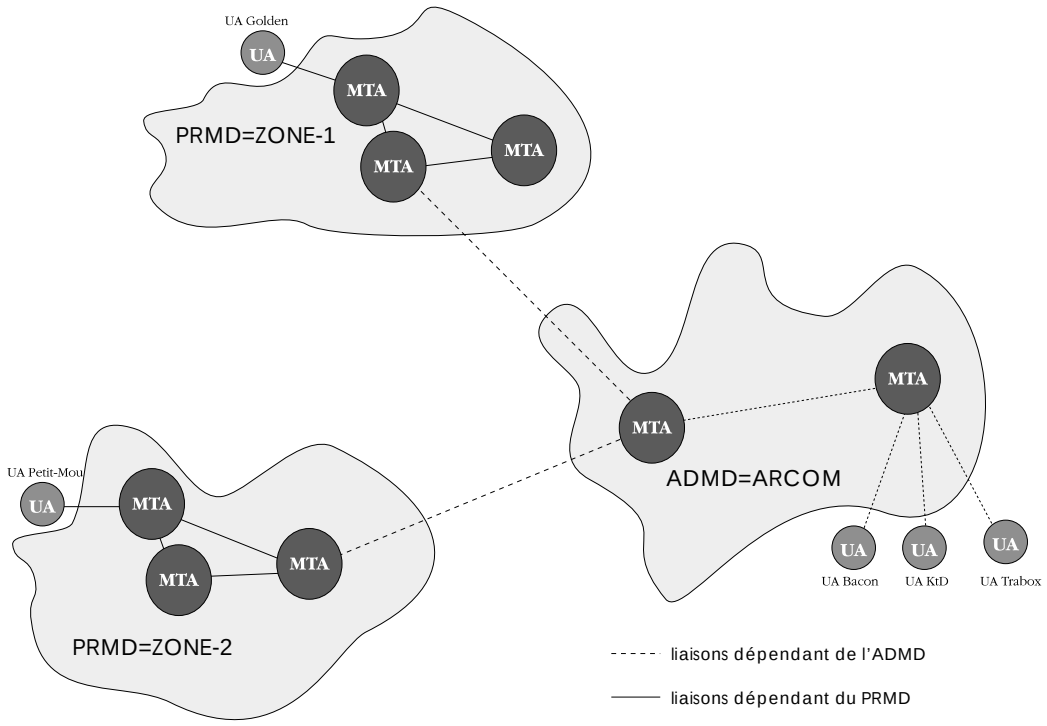
ODA : type de pièce utilisé pour des transferts de documents réalisés en traitement de texte.

Nationally defined : permet le support de normes définies au niveau national. Peu ou pas utilisé.

Bilat : *Bilaterally defined*. Pièce de type non défini, ou plutôt défini aux deux extrémités du transport selon des règles propres. Ces pièces sont transportés sans conversions aucune comme du binaire.

A noter la souscription d'un abonnement X.400 auprès d'une administration contient la description des types de pièces pouvant être traitée. Cela permet, le cas échéant, de faire au niveau de l'administration les conversions nécessaires. Cette remarque n'est, bien entendu, valable que pour les pièces de type texte. Il existe une indication précisant que le message ne doit en aucun cas être traduit d'un alphabet à un autre; dans ce cas, il n'est pas délivré si le destinataire ne peut pas le lire et un "non delivery" est retourné à l'expéditeur.

L'implantation nationale dépend du fournisseur de service. En Suisse, c'est l'entreprise des PTT qui fournit le service de messagerie X.400 sous le nom d'ARCOM. Les boîtes aux lettres peuvent se trouver soit aux PTT (cas dans l'exemple ci-dessous de KtD, Bacon et xxxx), soit sur des MTA privés se trouvant dans des domaines de gestions privés (PRMD) tels que zone-1 ou zone-2.



Dans l'exemple ci-dessus, on aurait des adresses du type suivant :

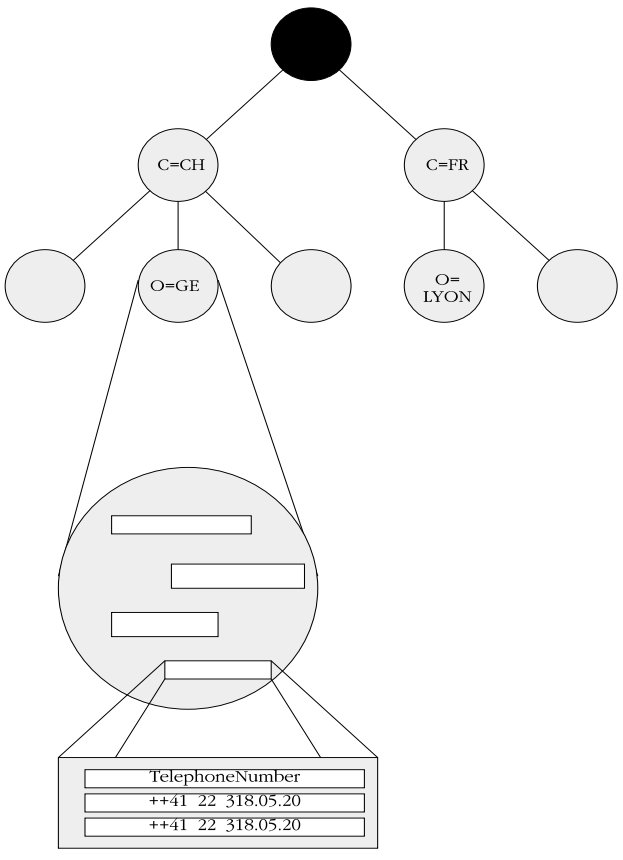
pour KtD : C=CH;A=ARCOM;S=KtD
pour Bacon : C=CH;A=ARCOM;S=Bacon
pour Trabox : C=CH;A=ARCOM;S=Trabox
pour Petit-Mou : C=CH;A=ARCOM;P=ZONE-2;S=Petit-Mou
pour Golden : C=CH;A=ARCOM;P=ZONE-1;S=Golden

18.- Les annuaires

Les annuaires sont très souvent considérés comme étant partie intégrante des messagerie. S'il est vrai qu'une messagerie avec un carnet d'adresse (parfois nommé annuaire) est un tout indissociable, il ne faut pas inclure l'annuaire dans la messagerie, à plus forte raison la norme X.500 à la norme X.400.

La norme X.500 définit un service de mise à disposition d'informations en tous genre, qu'il s'agisse de numéros de téléphone, d'horaires de train, d'adresse de serveurs de fichiers ou d'adresses de courrier électronique.

La base de donnée des informations dans un contexte X.500 est une base de données hiérarchique composée d'objets, eux-même ayant des attributs de nom, de type et de valeur donnée. Cette arborescence est mondiale et se nomme le "DIT" ou "Directory Information Tree"



La base de donnée est hiérarchique et chaque a une racine

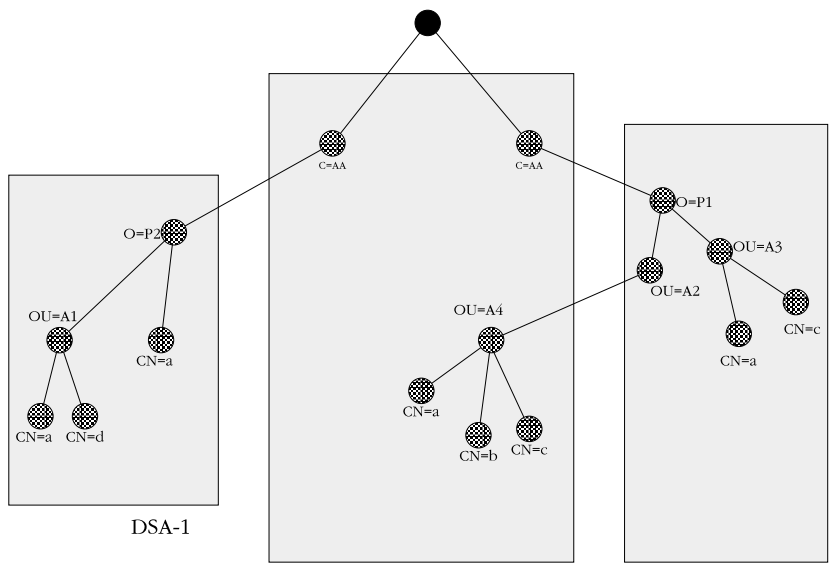
Chaque entrée a un nom unique si l'on considère toute l'arborescence que l'on appelle

Chaque entrée possède des attributs

Chaque attribut a un nom, un type et un ensemble de valeurs

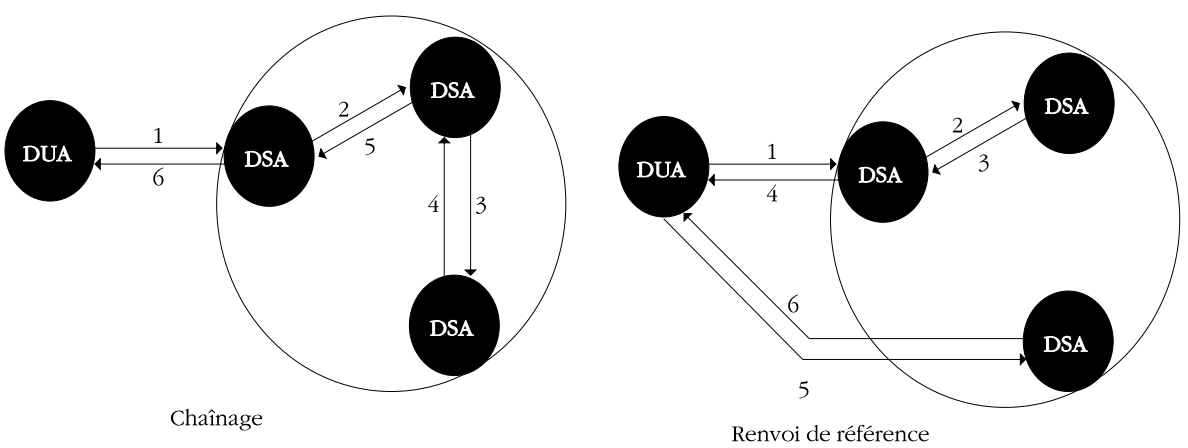
Le DIT s'étend mondialement, et chaque noeud est placé sous une autorité administrative, qui gère elle-même l'attribution des noms dans son domaine. Pour la Suisse, c'est Switch qui gère le domaine.

Le DIT est donc une vue logique et la base de donnée où sont stockées les informations est indépendante du DIT. La vraie structure, qui est répartie sur plusieurs machines (base de donnée distribuée, éventuellement jusqu'au niveau de chaque organisation) est totalement cachée à l'utilisateur. La consultation est l'opération la plus fréquente, la modification est très rare. On admet donc une certaine incohérence provisoire de la base en créant des "caches" ou antémémoires sur des machines plus rapidement accessibles. Cela permet d'évacuer la plupart des problèmes liés à une grande base de données distribuée.



Dans cet exemple, on voit qu'un DSA peut très bien servir à plusieurs niveaux et que les entités intermédiaires pourraient être sur un autre DSA.1

La base de donnée est formée d'un ensemble de DSA (*Directory System Agent*) qui communiquent entre eux au moyen du DSP (*Directory System Protocol*) soit directement, soit par DSA interposé. L'utilisateur, quant à lui, utilise un DUA (*Directory User Agent*) selon le DAP (*Directory Access Protocol*) pour accéder au DSA. Le DUA peut contacter n'importe quel DSA. Si ce dernier ne peut fournir l'information localement, il va soit contacter un autre DSA pour obtenir l'information (on parle alors de chaînage des requêtes), soit procéder par renvoi de référence et c'est le DUA qui contactera l'autre DSA.



19.- Centre informatique et infocentre

De plus en plus, les entreprises font la différence entre centre informatique et infocentre, qui regroupent deux notions bien différentes. Il existe des cas où la différence n'est pas très bien marquée. Mais voyons d'abord une définition de ces deux notions :

Un *centre informatique* est un centre qui regroupe des ressources informatiques et/ou de télécommunication et qui offre aux utilisateurs des ressources de temps de calcul, de stockage disque, de télécommunications. Il ne dispose de personnel que pour l'exploitation et la gestion des opérations systèmes (développements système, administration des ressources centralisées, etc.). Le développement des applicatifs tournant sur le système central sont le fait d'une équipe de développeurs et devrait être séparé de l'exploitation.

Un *infocentre* est une cellule de conseil et de mise en place des logiciels chez les "clients" et se trouve surtout dans un contexte d'informatique fortement décentralisée. Un infocentre gérera, le cas échéant, aussi les télécommunications dans l'optique de mises à jour des logiciels par télédéchargement lors de télémaintenance.

Le cas idéal, si les deux doivent exister, est de séparer la partie infocentre de la partie centre informatique, et d'avoir, le cas échéant une cellule de développement en plus. Le grand danger en cas de regroupement est d'avoir une mauvaise gestion des ressources, surtout humaine, les développeurs commençant de plus en plus à faire du conseil ou de l'exploitation et vice-versa.

Le centre informatique

Voyons maintenant quelques considérations concernant le centre informatique, depuis la décision de l'implanter jusqu'à son exploitation. Ce processus peut prendre de plusieurs mois à quelques années. Ces considérations touchent tant le personnel que la structure des locaux.

Première étape : définition des besoins

La première étape indispensable est la définition des besoins d'application (à savoir quels projets vont être mis en route) et de définir de quel matériel (puissance de calcul, mémoire, stockage de masse, télécommunication). Il faudra en outre définir de suite quelle politique d'exploitation est prévue pour la phase des tests et du développement : même machine avec répertoires fortement cloisonnés, machines séparées, exploitation de test en nocturne, etc.). En outre, le mode de travail dégradé en cas de panne partielle du matériel doit être défini : cela peut par exemple nécessiter d'avoir un second système de configuration environ identique disponible et d'avoir des techniques de basculement dûment documentées.

Une grande difficulté dans cette phase est l'évaluation des besoins réels : les utilisateurs ne se rendent jamais compte, à l'avance, des ressources dont ils auront besoin, voire même plus tard lorsque l'application est en exploitation.

On peut ainsi tirer un premier bilan des ressources nécessaires. Il importe de prévoir une installation modulaire, facilement extensible et qui n'hypothèque pas l'avenir : ne jamais choisir, par exemple, une machine haut de gamme, mais toujours viser en bas ou au milieu de la gamme, même si la solution est légèrement plus chère au démarrage. De plus, une première planification d'exploitation doit être faite, afin de savoir si l'on peut répartir dans le temps les grandes demandes de temps calculs et de ressources mémoire et stockage de masse.

On peut définir des catégories de besoins ou d'utilisateurs :

1.- type d'utilisateur PC utilisant des ressources centrales selon les besoins (time sharing, batch, transactionnel, temps réel). Il faut en outre savoir si c'est un développeur ou un utilisateur
2.- type d'utilisateur transactionnel : un moniteur gère des utilisateurs qui font les mêmes genres de transactions (moniteur ayant plusieurs claviers d'utilisateurs)
3.- temps réel : attribution du temps machine commandé par des événements extérieurs, notion de temps de réponse et de délais plus critiques (conséquence : le matériel doit disposer des périphériques qui détectent les événements et de la puissance de calcul de réserve suffisante pour prendre en charge les événements)

La confidentialité (accès à la machine, aux fichiers) sera un autre point important de la définition des besoins et la consolidation des données ne sera pas non plus négligée dans cette phase, tant sous l'aspect validation que sauvegarde, car cela peut avoir des incidences directes sur la configuration.

Enfin, la sécurité vue sous l'aspect "options" des locaux (feu, alimentation électrique, accès, catastrophes) devra être évaluée. Jusqu'où veut-on aller pour cette sécurité, sachant que plus on en veut, plus c'est cher ! (l'augmentation est exponentielle).

L'accès pour les livraisons, le stockage du papier et des fournitures (disques, bandes magnétiques, etc) ne devra pas être négligé.

On peut alors procéder à la

Deuxième étape : l'évaluation et la demande budgétaire

Elle se fera sur la base de configurations types répondant aux critères vus plus haut, et ce avec deux ou trois fournisseurs potentiels.

Troisième étape : établissement du cahier des charges

Il est souvent difficile et long à réaliser. Il importe de faire une formulation claire, précise et univoque, il doit être complet et surtout ne pas être fixé sur un constructeur ou une firme donnée.

Quatrième étape : l'appel d'offres

L'appel d'offres à proprement parler peut alors être lancé. Voici un certain nombre de points qu'un tel document devrait comporter :

- le cahier des charges (en annexe ou consultation libre),
- les indications budgétaires,
- les délais de reddition des offres et d'installation, respectivement de mise en service,
- les tests de plausibilité (prêt de machines pour les PC et test ou démonstration sur un site pour les grosses machines),
- les notions contractuelles (achat, leasing, contrat d'entretien),
- diffusion de l'appel d'offre : par envoi aux fournisseurs potentiels de la place et annonce dans la feuille d'avis officielle (FAO). Cette manière de faire permet d'atteindre, en théorie du moins, tous les dirigeants d'entreprise éventuellement concernés,

Quelques points sont à préciser lors de l'envoi de l'appel d'offres :

- la réalisation de l'offre et, le cas échéant, les tests de plausibilité sont à la charge du soumissionnaire,
- le contenu de l'appel d'offres, notamment du cahier des charges, reste confidentielle. Le maître d'oeuvre s'engage à en faire de même avec les offres,
- préciser que les critères d'évaluation sont du ressort exclusif du groupe d'évaluation, dont la liste des membres n'est pas publiée, ceci pour éviter les contestations et prises de contact qui pourraient fausser les décisions,
- la manière de répondre à l'appel d'offres, afin d'être sûr que les offres seront comparables et ne nécessiteront pas de longs échanges épistolaires ou verbaux pour en permettre l'évaluation,
- l'acceptation, par le soumissionnaire, des conditions de l'appel d'offre par le fait qu'il y réponde,
- si des contrats types existent, tant pour l'achat que l'entretien, il est bon de les joindre aux documents afin que le soumissionnaire prenne position et signale les clauses qu'il voudra modifier,

Cinquième étape : évaluation des offres

Il convient d'établir une grille d'évaluation des offres avant d'en commencer le dépouillement, et de notamment bien définir quels seront les critères éliminatoires. De plus, le travail d'évaluation devra être fait par un groupe représentatif des différents aspects de l'utilisation, respectivement des utilisateurs prévus. Les travaux de la commission d'évaluation doivent rester confidentiels jusqu'à la reddition du rapport ou du choix au mandataire. En outre, il est préférable de ne pas donner le poids des critères aux soumissionnaires.

Sixième étape : planification des travaux

Les travaux préparatoires doivent être soigneusement planifiés. Il conviendra de tenir compte du fait qu'une exploitation parallèle sera peut-être nécessaire entre l'ancienne et la nouvelle installation. Dans les travaux préparatoires entrent :

- préparation du site (Entreprises de construction, PTT (lignes téléphonique commutées et louées), Services industriels (alimentation électrique) etc.); cette opération prend du temps !
- travaux informatiques préparatoires (préparation des jeux de tests, portage des applications existantes, etc.)

Septième étape : installation (hardware et software)

L'installation, tant hardware que software, sera faite par le soumissionnaire. Il s'agit de bien définir la limite à partir de laquelle l'entreprise elle-même continue l'installation de ses logiciels propres. La "passation des pouvoirs" se fera sous forme d'un point fixe où des tests d'acceptation, défini d'un commun accord lors de la conclusion du contrat, sont faits. Ces tests servent notamment à s'assurer que ce qui a été livré et installé est conforme au contrat. Afin de garder une réserve, on peut très bien convenir d'un payement en trois étapes par exemple : un tiers lors de la signature du contrat, un tiers à la livraison et installation du matériel et du logiciel, et le dernier tiers en fin des tests d'acceptation.

Huitième étape : Mise en place de l'exploitation

La mise en place d'un système informatique centralisé nécessite du personnel. Les fonctions suivantes doivent être prévue, même si elles sont cumulées sur un nombre de personnes restreint. On trouvera donc le personnel :

- d'exploitation (comprenant les opérateurs principalement),
- l'administrateur du système, qui définira notamment les procédure d'exploitation, attribuera les ressources et les quotas, etc.

- la cellule de développement (analyse-programmation et système)
- la cellule de conseil et documentation

Si ces deux dernières fonctions sont regroupées sur des personnes identiques, il conviendra de très bien définir les tâches et le pourcentage de temps attribué, faute de quoi l'aspect conseil et développement sera confondu et plus aucune gestion des priorités dans le travail ne sera possible.

Neuvième étape : la maintenance

La maintenance se décompose en deux grandes parties bien distinctes : la maintenance matérielle et la maintenance logiciel.

Si la maintenance matérielle est souvent remise sous forme de contrat à une entreprise (le fournisseur par exemple), elle nécessite une attention particulière sous l'aspect de la détection d'une part, sous l'aspect du délai d'intervention d'autre part. Le contrat d'entretien définira ces différents problèmes. Un délai d'intervention court (de l'ordre de quatre heures) à long (de l'ordre de deux à trois jours). Le délai d'intervention influencera très fortement et rapidement les conditions financières de l'accord, car il influence le nombre de personnes formées devant être de piquet chez le prestataire de la maintenance.

Un autre point à bien définir est celui du délai de panne maximum sans qu'un matériel de remplacement de qualité et capacité identiques soit remis en prêt. Le temps de reconfiguration de la machine fait partie, dans ce calcul, du temps de panne ou de reprise si cette notion est définie contractuellement.

La reprise est souvent conditionnée par les application de l'entreprise. Il ne faut donc pas oublier que du personnel d'exploitation devra être présent à ces moments.

Le personnel d'exploitation et l'administrateur devront en outre mettre en évidence les bugs éventuels au niveau système d'exploitation et utilitaires généraux, appliques les mesures de sécurité prévues, assumer un piquet en cas de besoin. En outre, des mesures de charge du système en vue du maintien des performance avec l'augmentation de la charge est une mesure préventive importante. Or toutes ces fonctions nécessitent des ressources et, par conséquent, générer des frais d'exploitation.

Gestion des frais d'un centre informatique

Le centre informatique est en général considéré comme une charge pour l'entreprise. Il s'agit de comptabiliser ces charges d'une manière ou d'une autre. On peut envisager, par exemple, les procédés suivants :

1.- On le considère comme charge sans détailler les prestations. Le coût passe dans les frais généraux au niveau de l'entreprise.

2.- Le centre informatique n'est pas un centre de frais pour l'entreprise, mais une entreprise dans l'entreprise qui facture, ne fusse que pro format, les services qu'il rend au reste de l'entreprise. Une partie, tel que par exemple les locaux ou la fourniture d'énergie, peut toutefois rester comme frais généraux de l'entreprise.

La seconde variante présente l'avantage de responsabiliser les chefs de services sur l'aspect de l'explosion des charges. Ils réfléchiront ainsi mieux si une tâche peut, doit ou surtout ne doit pas, être informatisée.

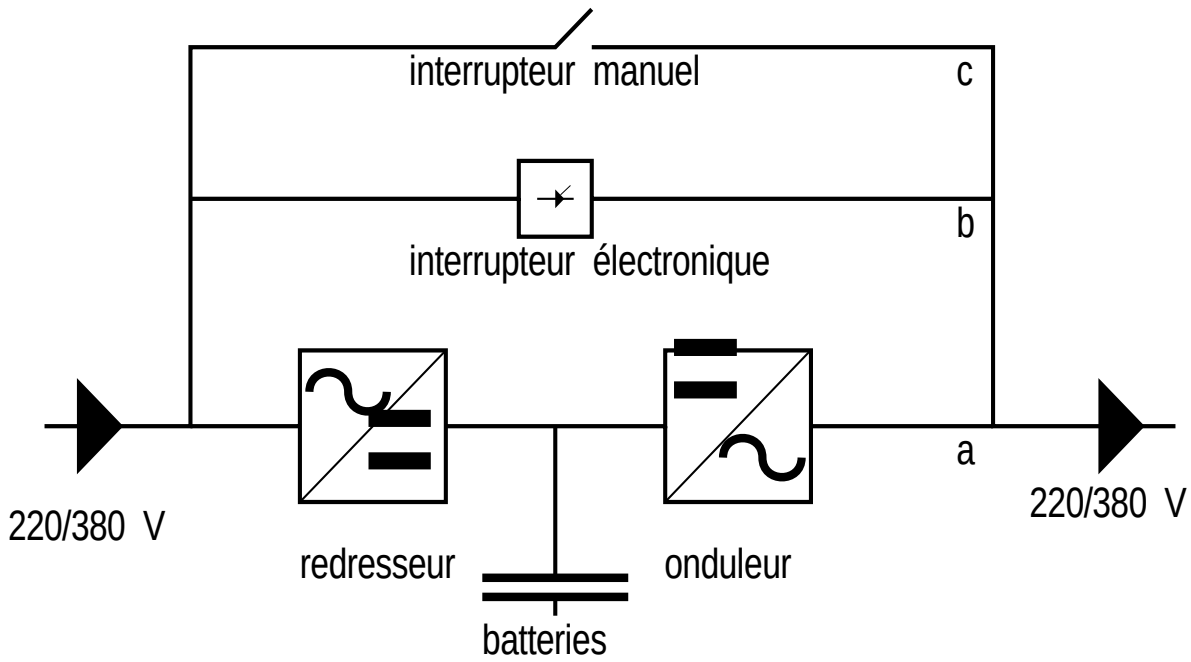
Equipement des locaux d'un centre informatique

Ci-dessous quelques points auxquels il faut avoir réfléchi lors de l'installation ou de l'équipement d'un centre informatique. Cette liste n'est pas exhaustive, mais soulève certains points auxquels il faut avoir pensé, même si cela conduit à la décision de ne rien entreprendre sous l'un ou l'autre des aspects.

- climatisation : la température ambiante doit rester stable entre 18^{o} et 24^{o}. Bien plus que d'avoir une température bien précise, il importe que les variation soient faible, en général au maximum de l'ordre de 0.5^{o} à 1^{o} par heure. L'humidité relative de l'air est également importante : au minimum 40%, idéal entre 40 et 60%. Si l'air est trop sec, on aura des ennuis d'électricité statique, et ce surtout au niveau des imprimantes. Il convient, lors des calculs, de tenir compte du dégagement calorifique des machines et des hommes travaillant dans ce local.
- évacuation de l'air chaud : il va aussi falloir penser à la manière dont la chaleur produite sera évacuée (échange calorifique avec de l'eau (interdit pour des débits moyens à grands), aérorefroidisseurs)
- accès aux locaux : il faut penser aux livraisons (des machines par exemple...) et aux livraisons ultérieures (papier : accès avec un transpalette possible) Attention : si un système de contrôle d'accès est mis en place, il ne faut pas que ces accès de livraison soient une faille dans le système. Il en va de même des sorties de secours).
- faux planchers : l'idéal est d'en avoir partout dans la zone informatique. Le faux plancher sert à tirer proprement des câbles (courant fort et faible), mais sert aussi au conduites d'air refroidi. Une hauteur de 25 cm à 30 cm est raisonnable dans la plupart des cas.
- faux plafonds : sert à l'évacuation de l'air chaud

- chemins de câbles : les prévoir dans les faux plancher, dans les faux plafonds, en séparant le courant fort (220 - 380 Volts - alimentation électrique) du courant faible (50 volts - téléphone, données) afin d'éviter des perturbations
- canaux d'allège : canaux se fixant aux murs, à environ 1 m du sol, dans lesquels on installe le courant électrique et les câbles des réseaux. Attention, il faut les prévoir assez large !
- feu : trois niveaux sont envisageables : rien, détection, extinction. La détection peut être raccordée à une centrale d'alarme ou aux pompiers. Elle n'est utile que si le système permet l'intervention rapide d'une équipe d'une part, qu'elle ne donne que de vraies alarmes d'autre part. L'extinction peut se faire de plusieurs manières : la meilleure sous l'aspect efficacité est certainement le halon (halon 1301, bromotrifluorométhane, CF_3Br, utilisé souvent en installation fixes, ou halon 1211, bromochlorodifluorométhane, CF_2ClBr, plutôt utilisé en extincteur portatif) Ces gaz halons font partie de la game des gaz CFC et posent problème sous l'aspect écologique (couche d'ozone). Ils seront interdits dans les nouvelles installations depuis le début des années 90 et une ordonnance devrait en interdire tout usage dès 1997. Les autres méthodes sont à base de gaz carbonique ou d'eau (sprinklers).
- alimentation électrique : il s'agit de déterminer quelles sont les machines, respectivement périphériques, qui doivent rester actifs en cas de coupure d'électricité, respectivement qui doivent être informés d'une chute imminente de tension. A cet effet, on met un onduleur sur l'alimentation avec des batteries qui agissent en tampon. La durée d'alimentation en cas de coupure réseau dépend de la consommation d'une part, de la charge et capacité des batteries d'autre part.
  Sont en général connectées sur l'alimentation secourue les machines suivantes : les unités centrales, les disques, les consoles maîtresses, les télécommunications, le contrôle d'accès et les alarmes, un éclairage minimum.

Voici le schéma de principe d'un onduleur :



Le circuit marqué "a" représente le circuit normalement en exploitation : les batteries sont continuellement rechargées et forment un tampon pour palier à des microcoupures éventuelles de l'alimentation. Le circuit "b" est coupé en exploitation normale, mais permet une reprise immédiate en cas d'ennuis de l'onduleur ou de batteries déchargées. A noter que, dans ce cas, l'onduleur doit absolument être en phase avec l'alimentation. Enfin, le circuit "c" est une possibilité manuelle de mise hors service de l'onduleur et de l'interrupteur électronique.

- Les lignes de télécommunications, particulièrement celles qui utilisent le réseau loué, nécessitent en général des délais de commandes extrêmement longs (de quelques semaines à plusieurs mois, voire années si la liaison passe dans un secteur surchargé ou installé en aérien.
- Il faut prévoir aussi un lieu de stockage des sauvegardes. Il est important que le lieu soit différent de celui du centre informatique, afin que, en cas de catastrophe, tout ne soit pas perdu. Les backups journaliers peuvent être stocké dans le local machine, éventuellement dans un coffre anti-feu. Les backups hebdomadaire ou mensuels devraient être stockés ailleurs. Cette politique dépend énormément de la valeur des informations et sera différente dans un contexte d'enseignement et dans une banque par exemple.
  Le but est que la probabilité de destruction simultanée de l'endroit où se trouvent les données et la probabilité de destruction de l'endroit où se trouvent les backups soit nulle ou quasi-nulle.

Infocentre

L'orientation principale d'un infocentre est le conseil, respectivement l'aide aux utilisateurs. En principe, il n'y a pas de ressources centralisées ni de cellules de développement. Il est d'ailleurs préférable de séparer l'aspect infocentre du développement pour éviter notamment les petites modifications non testées dans des logiciels, et d'éviter des retards, les nouvelles applications n'étant pas prêtes, les développeurs ayant du répondre et conseiller les utilisateurs plutôt que de vaquer à leur travaux. Dans un infocentre, il y aura donc beaucoup de personnel et peu de matériel :

- matériel utilisé par le personnel (station de travail)
- matériel de télécommunication
- pas de ressources de calculs ou de stockage centralisée.

Les fonctions et qualités du personnel d'un infocentre sont principalement :

1.- l'assistance et le conseil aux utilisateurs
2.- l'ouverture d'esprit
3.- la patience
4.- "assistance sociale" pour, par exemple, consoler la secrétaire (ou même un développeur, ce qui est encore moins pardonnable) qui n'a pas fait ses backup et qui a écrasé un disque
5.- sécuriser l'utilisateur (qui prend souvent contact lorsqu'il est en situation de crise)
6.- la formation des utilisateurs sur les produits d'usage courant tels que traitement de texte, tableur, etc.
7.- la diffusion de l'information, voire la confection de modes d'emploi des produits d'usage courant
8.- enfin, dans un contexte micro-informatique, le problème des virus devra être traité et pourrait très bien être sous le contrôle de l'infocentre.

En outre, l'évaluation et le test de logiciels (ou nouvelles versions), voire de nouveaux matériels destinés aux utilisateurs dont on a la charge, fait partie du travail normal d'un infocentre. Dans ce cas, il faut faire la planification du changement de révision, avec établissement des procédures de conversion des données et d'installation des lociciels. L'installation des nouvelles versions peut se faire, lorsque le réseau le permet, à distance. C'est ce que l'on nomme la télémaintenance. (Dans ce cas, il ne faut surtout pas négliger l'aspect sécurité pour l'accès aux serveurs distant, afin de se prémunir contre des accès non autorisés).

Enfin, l'accès à un "host" peut nécessiter des logiciels particulier, qui font partie de ceux qui devront être testés et installés.

20.- Serveurs télématiques

télématique

La télématique, bien qu'existant depuis plusieurs décennies, n'a vraiment pointé son nez dans les foyer qu'au cours des années 80. Contraction de télé(phone, communication, ...) et d'informatique, elle porte surtout sur l'informatique à distance (téléinformatique), mais actuellement surtout dans le sens "grand public".

Les Français ont bien compris ce qu'est la télématique, puisque c'est eux qui ont commencé les premiers à mettre à disposition de tout un chacun des terminaux "minitel" et à mettre en place des serveurs et services.

Basé sur un point d'accès simple, taxé en fonction du temps de communication avec rétrocession au fournisseur d'information, les serveurs sont apparus très rapidement et le service "kiosque" 3615 a de nombreux partenaires.

Le vidéotex suisse est par contre beaucoup plus complexe de par sa structure : il nécessite une centrale qui joue le rôle d'un "front-end processor" pour les entrées-sorties d'une part, de chef de gare pour l'accès aux services d'autre part. C'est également elle qui se charge du problème de taxation des usagers.

Le système de tarification étant assez complexe et plutôt favorable aux PTT, plusieurs centrales privées ont été mises en service par des entreprises ou administrations. C'est le cas notamment de l'Etat de Genève (département de l'instruction publique).

21.- Tests de plausibilité

Les facteurs humains

Introduction

Les systèmes interactifs à temps partagé sont nés de la nostalgie de certains programmeurs, nostalgie du "bon vieux temps" où l'ordinateur était à l'entière disposition de son unique utilisateur qui pouvait ainsi utiliser toutes les ressources et tous les dispositifs du système selon les besoins de son travail. En fait, c'est le désir d'éliminer les barrières entre le programmeur et la machine ou, plus simplement, le désir d'établir de meilleures relations entre l'homme et la machine qui est à l'origine du développement des systèmes time-sharing. C'est ce que l'on nomme l'ingénierie des facteurs humains ou ergonomie.

Il y a essentiellement deux manières d'utiliser l'ordinateur lorsque l'on résout un problème. Dans la situation traditionnelle, l'ordinateur est utilisé pour résoudre un problème dont on connaît déjà une partie de la réponse, en ce sens que le problème est bien défini et la méthode à employer déjà connue. Mais de plus en plus l'utilisateur ne résout pas un problème dans ce sens mais cherche plutôt à définir son problème à l'aide de l'ordinateur. La machine est utilisée pour essayer d'établir la nature et l'étendue du problème et ensuite, par essais successifs, le type de méthode ou la classe d'algorithmes pouvant être utilisés pour le résoudre. Pour ce type d'activités, qui implique une interaction, un dialogue hommes et machines doivent être en contact étroit.

Objectifs pour les concepteurs de systèmes interactifs

La diversité des situations où les systèmes interactifs peuvent être utilisés rend difficile la définition d'objectifs universels. Citons-en quelques-uns en vrac tirés de la litérature citée en bibliographie :

pour Pew, Rollins :

- connaître la population d'utilisateurs,
- répondre clairement et de manière con sis tan te,
- dégager une représentation des connaissances de base de l'utilisateur,
- adapter les messages aux besoins de l'utilisateur,
- donner aux utilisateurs toutes les occasions de corriger leurs propres erreurs,
- promouvoir la valeur personnelle de l'utilisateur individuel,

pour Hauser :

- minimiser l'effort de mémorisation,
- sélection plutôt qu'entrée complète au clavier,
- utilisation de noms plutôt que de nombres,
- comportement prévisible,
- accès à l'information concernant le système,
- opérations optimisées,
- exécution rapide des opérations banales,
- inertie de l'écran (pas trop de changements à la fois),
- mémoire musculaire (clavier bien étudié),
- réorganisation des paramètres de commandes,
- ingénierie tenant compte des erreurs possibles,
- bons messages d'erreurs,
- éviter les erreurs banales,
- action réversible,
- redondance,
- intégrité des structures de données,

pour Gaines et Facey :

- introduction par l'expérience,
- réaction immédiate,
- consistance et uniformité,
- éviter la causalité,
- interrogation par degrés successifs,
- choix entre opérations séquentielles et parallèles,
- système observable et contrôlable,

pour Cheriton : simpliste :

- projeter une image virtuelle du système qui soit naturelle et pas compliquée; réponse rapide et intelligible aux commandes de l'utilisateur,
- contrôle par l'utilisateur de toutes les actions initialisées ou contrôlées par lui,
- flexibilité dans la structure des commandes et des tolérances en cas d'erreurs,
- stabilité par la capacité de détecter les difficultés de l'utilisateur et de l'aider à reprendre un dialogue correct,
- protection de l'utilisateur face aux erreurs coûteuses ou aux accidents tels que la destruction d'un fichier par surécriture,
- autodocumentation par des commandes et des réponses du système compréhensibles par elles-mêmes. Des explications, de la documentation et des outils éducatifs sur le système font partie du système même,
- fiabilité ne conduisant pas à des erreurs non détectées dans la communication homme-machine,
- modification possible par l'utilisateur lui permettant, s'il est avancé, de personnaliser son environnement.

Evaluation des performances

Généralités

Historiquement, seules les performances hardware étaient évaluées. L'attention était portée sur la taille des mots, le cheminement des données, le nombre d'adresses par instruction, l'orientation caractère par rapport à l'orientation nombre. Puis aussi les performances du stockage de masse, des canaux d'entrées-sorties et des unités périphériques. Puis encore : look-ahead, interleaving, système d'exploitation, programmation, vitesse d'assemblage et de compilation, package de télécommunication, etc.

Introduction

Il y a trois buts dans l'évaluation des performances :

- techniques d'évaluation en vue d'un achat (implique l'existence du matériel à tester),
- projection des performances en vue de la création d'un nouveau système ou de la modification des performances d'un système existant,
- évaluation des performances d'un système existant, l'impact de la charge d'un système et l'évaluation d'une nouvelle configuration.

Considérons, à titre d'exemple, certains tests d'évaluation :

- temps de cycle et d'addition : comparaison des temps CPU, d'addition et d'accès à la mémoire; défauts : organisation de la machine ignorée, inutile sur une machine orientée caractères, ne tient pas compte du cheminement des données, l'instruction d'addition ne représente même pas le hardware, ignore le software,
- mélange d'instructions : utilisation de la fréquence d'exécution des instructions, par exemple les applications commerciales tiennent compte des instructions d'édition, de consultation de table et de déplacement de données; pour les applications scientifiques, tous les calculs. Défaut de cette méthode : taille des mots, cheminement des données, logique d'adressage, software, entrées-sorties sont des notions ignorées.

- noyau de programmes : il s'agit d'un ou plusieurs programmes typiques, partiellement ou complètement codés dont on prédit le temps d'exécution. Ces mesures sont basées sur les temps d'exécution fournis par le constructeur. A la fois des algorithmes simples et compliqués sont codés qui vont des systèmes de calculs de paie jusqu'aux inversions de matrices. Auerbach EDP Standards donne un choix très vaste de tels programmes : ces programmes peuvent inclure plus de paramètres pour mieux tenir compte du choix de l'ensemble des instructions. Mais par contre, le cheminement des données, la multiprogrammation et le système d'exploitation sont difficiles à estimer. Cette technique est particulièrement utile pour accéder aux performances hardware. Des indications sur le software peuvent être obtenues en examinant le code objet lors de la compilation.
- modèles analytiques : il s'agit d'une représentation mathématique d'un système. Cette technique est fréquemment utilisée pour tester les performances d'un composant particulier du système, par exemple la gestion des accès aux disques. Cette technique ne montre pas l'impact du système d'exploitation et du software.
- benchmarks : c'est un ou plusieurs programmes existant, codés dans un langage spécifique et exécutés sur la machine à évaluer. Une série d'exécutions de benchmarks peut montrer des différences au niveau de l'organisation machine et évaluer les performances d'entrées-sorties, de stockage de masse aussi bien que la variété du jeu d'instructions. Ils contiennent donc des considérations software, de vitesse de compilation et d'exécution et des comparaisons sur la vitesse d'exécution du code. Ils peuvent aussi être testés sous le contrôle d'un système d'exploitation.

Simulation

Il y a deux types de simulation pour évaluer les performances des ordinateurs :

- un modèle de simulation pour les opérations actuelles d'un système. Un calendrier des événements est maintenu et des probabilités sont utilisées pour décrire la performance des composants à évaluer.
- le deuxième type de simulation utilise empiriquement des données correspondant à une configuration spécifique.

Le premier type de simulation a été utilisé pour le design de systèmes nouveaux alors que le deuxième type a été utilisé pour simuler sur des machines existantes le comportement de nouvelles machines généralement compatibles entre elles.

Les ordinateurs diffèrent tellement dans leur organisation que chaque simulation doit être élaborée pour une machine individuellement. La simulation est particulièrement avantageuse pour projeter les performances de systèmes nouveaux.

Contrôle des performances

Il s'agit d'une méthode qui récolte des données sur la performance d'un système existant. Elle est généralement utilisée pour localiser des goulots d'étranglement limitant les performances d'une machine, lorsqu'on reconfigure le hardware ou lorsqu'on améliore le software existant. La méthode est utile aussi pour déterminer le pourcentage de jobs orientés calculs pour déterminer de nouvelles priorités. Par exemple, si des statistiques montrent une portion élevée de temps de compilation comparé au temps d'exécution, une installation nouvelle devra prévoir des compilateurs plus rapides qui produisent un code plus lent. Les techniques de contrôle de performance hardware utilisent des "boîtes noires" qui mesurent certains paramètres comme le temps CPU et le temps d'E/S combinés avec les temps d'attente et d'overlap. D'autres paramètres comme le temps total utilisé par le système, le temps total de compilation, le temps d'accès aux disques, etc. Généralement, c'est le système d'exploitation, lui-même, qui collecte ces statistiques. Le cas extrême est celui où toutes les modifications d'état du CPU sont répertoriées (attention à la dégradation des performances). Les données récoltées sont ensuite triées pour montrer les différentes fréquences d'utilisation du CPU, des E/S, de la pagination, de la mémoire et des périphériques. Elles permettront de déterminer ce qui est à modifier, améliorer, optimiser dans un système.

22.- Petit glossaire

Ce petit glossaire explicite des termes utilisés dans ce fascicule. La plupart de ces définitions sont tirées du Some-Bit 60, Petit Glossaire de Termes Informatiques, publié par le CCEES (Cf bibliographie), d'autres de glossaires réalisés par Jérôme Ponitz, CCI, dans le cadre de diverses études.

10Net : Nom d'un logiciel de gestion de réseau développé par DCA (Digital Communication Associate), repris depuis par TIARA. Ce réseau local, choisi par le Département de l'Instruction Publique Genevois en 1984 déjà, présente l'avantage d'être simple et transparent à l'utilisateur. Il permet de monter des serveur dédicacé ou non, qui utilisent une structure DOS pour la gestion des fichiers du serveur. Il utilise une couche de transport NETBIOS, qui peut être du même fournisseur ou non. A noter que le DIP dispose d'une licence de site pour ce produit.

Absolu : se dit du code objet lorsque les adresses sont définitives. Le programme devra donc être chargé à l'adresse spécifiée.

Accès : caractérise la manière de référencer une information se trouvant en mémoire ou sur un support externe. L'accès peut être soit séquentiel, soit direct, soit séquentiel indexé (voir ces termes).

Adresse : caractérise l'emplacement physique où se trouve stockée une information. On peut comparer l'adresse informatique au numéro postal d'une localité. L'adresse est surtout utilisée pour référencer une position mémoire, un enregistrement sur disque ou sur floppy.

ADMD : *Administration Management Domain*. Ensemble de MTA (voir ce terme) constituant, en matière de messagerie éléectronique, un domaine public d'administration. En Suisse, à ce jour, il existe deux domaines : ARCOM, domaine des PTT, et IBMX400.

ANSI : *American National Standards Institutes*. Principal organisme de normalisation aux Etats Unis. Il regroupe plus d'un millier d'entreprises et sociétés commerciales. L'ANSI est membre de l'ISO pour les USA.

Aléatoire : se dit parfois pour désigner l'accès à des fichiers ou à des périphériques tels que disques pour bien indiquer que l'on a accès à n'importe quel enregistrement sans devoir lire les précédents. Se dit également de nombres pris au hasard.

API : *Application Programming Interface*. Interface de programmation. C'est un ensemble de règles de programmation et de point d'entrées dans un applicatif pour des programmes utilisateurs. Les API permettent de programmer dans un langage en utilisant les ressources d'un applicatif, messagerie par exemple.

ARPA : *Advanced Research Project Agency*.

ARPANET : *Advanced Research Project Agency NETwork*. Réseau à commutation de paquets en mode datagramme du DOD et des universités américaines. Ce réseau est le précurseur des réseaux TCP/IP.

Article : Elément d'information formant un tout en vue d'un traitement informatique. Un article peut se trouver, dans un fichier de données, sur un - cas le plus fréquent - ou sur plusieurs enregistrements physiques ou logiques.

ASCII : *American Standard Code for Information Interchange*. Code permettant de représenter les caractères en machine. Le code ASCII se compose de l'alphabet, des chiffres, de signes spéciaux tels que point, virgule, parenthèse, etc, et de caractères de contrôle (bell, enq, ack, eot, etc). ASCII est l'abréviation de "American Standard for Communication Interface Interchange". Les autres codes courants sont le BCD et l'EBCDIC. Le code ASCII se compose de 128 caractères, ce qui nécessite 7 bits pour les représenter. En général, on prendra un huitième bit comme bit de parité pour vérifier la validité des sept autres.

Assembleur : L'assembleur est un langage défini pour un ordinateur donné. En assembleur, les instructions utilisées sont celles du langage machine remplacées par des mnémoniques. La programmation en langage assembleur est parfois fastidieuse car chaque petit détail doit être indiqué, mais elle a l'avantage de permettre l'utilisation de toutes les ressources du hardware, ce qui est rarement le cas des langages évolués.

Asynchrone : Transmission série entre terminaux et/ou ordinateurs ayant lieu caractère par caractère, le temps entre deux caractères n'étant pas défini. Chaque caractère est précédé d'un start-bit et suivi d'un ou deux stop-bits, ce qui signifie que, en ASCII, il faudra 10 à 11 bits par caractère. L'avantage des transmissions asynchrones réside dans la simplicité de la méthode (le caractère est envoyé dès que la touche est appuyée).

ATM : *Asynchronous Transfert Mode*. Protocole de communication par cellule de 53 bytes. Permet des liaisons orientées connexion, c'est-à-dire à circuits virtuels et permet des liaisons isochrones (respect du temps de transmission constant entre départ et arrivée, ce qui est important dans le transport de la voix). Les vitesses de transmission sont de 34 Mbits par seconde, 155 Mbits par seconde en version SONET, voire 600 Mbits par seconde, en transport fibre en général.

BAL : *Boîte aux lettres*. Terme souvent utilisé pour désigner une boîte aux lettres électronique dans le contexte de la messagerie.

Batch : Technique de passage de travaux d'utilisateurs enchaînés séquentiellement. Chaque "job" n'est soumis qu'une fois qu'il est complet, c'est-à-dire que toutes les données nécessaires au traitement y ont été inclues. Il peut contenir des compilations et/ou des exécutions, y compris les programmes sources pour des compilations ou les données pour des exécutions. Le terme français consacré est : traitement par lots.

Bauds : Caractéristique de toute transmission série. C'est le nombre de bits transmis par seconde. Le 110 bauds correspond à environ 10 caractères par seconde, le 600 bauds à environ 60 caractères par seconde.

BCD : Code de représentation interne de caractères. BCD est l'abréviation de "Binary Coded Decimal". C'est le code utilisé sur certaines grandes installations, telle la HB66/20P du Centre Cantonal d'Informatique (CCI).

Bit : de l'anglais "binary digit" : chiffre binaire ne pouvant prendre que deux valeurs : 0 ou 1, allumé ou éteint, courant ou pas de courant, sens positif ou négatif.

BNF : BNF est l'abréviation de Backus-Naur Form. Cette notation est surtout utilisée pour la définition de la syntaxe de langages; tel est le cas de l'Algol.

Bridge : Passerelle reliant deux ou plusieurs réseaux locaux en filtrant le passage des paquets sur la base des adresses MAC au niveau 2 du modèle ISO-OSI (adresse des cartes éthernet par exemple). Lorsque le bridge constate qu'il s'agit d'un paquet dont l'émetteur et le destinataire sont de part et d'autre ou qu'il s'agit d'un message de type broadcast, il recopie le paquet sur l'autre brin, sinon il le filtre et ne le recopie pas. Les bridges sont utilisés sur les grands réseau pour limiter le trafic, et par conséquent les collisions dans un contexte CSMA/CD.

Broadcast : se dit d'un message qui n'a pas de destinataire unique, mais qui va à chaque poste du réseau. Utilisé pour s'annoncer sur le réseau ou pour rechercher un service lorsque l'on ne sait pas quel est le poste qui le fournit.

Bus : Ensemble de lignes électriques reliant différents éléments d'un ordinateur. Les lignes du bus se décomposent en lignes d'adresse, lignes de données et lignes de contrôle. Le bus relie en général l'unité centrale avec la mémoire et/ou les périphériques; il peut aussi relier entre elles 2 ou plusieurs unités centrales.

Byte : Ensemble de 8 bits. On l'utilise surtout pour la représentation d'un caractère. (Cf aussi octet).

Canal : Contrôleur sophistiqué permettant à un périphérique ou à un groupe de périphériques de transmettre à haute vitesse de l'information plus ou moins directement à la mémoire sans qu'il y ait intervention de programme entre l'initialisation du transfert et sa fin.

Capacité : Quantité d'information pouvant être stockée simultanément sur un support interne ou externe. On parle de bytes ou de mots (voir ces termes). L'unité utilisée est le "K" (1 K = 1024 = 2 puissance 10) ou le "méga (million).

Caractère : Signe pouvant éventuellement être imprimé. Le jeu de caractères se compose de l'alphabet, des chiffres et des signes spéciaux. Chaque caractère est représenté en machine par un code. Le code le plus souvent utilisé est le code ASCII.

CCITT : *Comité Consultatif International Télégraphique et Téléphonique*. Le CCITT fait partie de l'UIT (Union Internationale des Télécommunications) dont le siège est à Genève.

CEN : *Comité Européen de Normalisation*. Comité regroupant 18 pays européens de la CEE et de l'AELE éditant des normes dans tous les domaines couverts par l'ISO, sauf l'électricité et l'électronique. L'application des normes CEN est obligatoire pour les pays membres, contrairement aux normes ISO.

CENELEC : *Comité Européen de Normalisation ELECtronique*. Même chose que le CEN, mais pour la partie électronique.

Circuit : terme souvent utilisé en communication pour désigner une communication. En téléphonie, on parle de commutation de circuits, alors que les réseaux de commutation par paquets de type X.25 utilisent la notion de circuit virtuel.

Circuit intégré : Ensemble de circuits élémentaires regroupés dans un boîtier pouvant avoir jusqu'à 60 contacts extérieurs. La taille du boîtier est en général de 3 mm d'épaisseur, 5 à 20 mm de largeur et 15 à 30 mm de longueur; il peut contenir de 1'000 à 300'000 composants (transistors, condensateurs, résistances, etc). On utilise parfois le terme anglais : "chip".

Compilateur : Programme servant à traduire un programme utilisateur d'un langage évolué tel que le Fortran ou le Cobol en un langage objet ou machine.

Compilation : Action consistant à compiler ou à traduire un langage évolué en un langage objet, assembleur ou machine.

Computer : Terme anglais signifiant ordinateur.

Concentrateur : Appareil ou interface permettant de communiquer avec plusieurs périphériques au moyen d'une seule ligne physique. Le concentrateur est surtout utilisé pour la gestion des terminaux.

Console : Unité périphérique servant au dialogue entre ordinateur et utilisateur. En général, il s'agit soit d'un panneau avec des indicateurs de bon fonctionnement ou d'anomalie, soit d'une machine à écrire ou d'un écran. S'il s'agit de celle permettant le contrôle du système, on parlera de console maîtresse.

CP : Abréviation de "*Central Processing Unit*" (Cf. ordinateur).

CRC : *Cyclic Redundancy Check*. Code de redondance cyclique. Le CRC est utilisé pour se prémunir contre les pertes d'informations ou leur altération en créant une redondance limitée permettant de garantir la non-altération du message. Les CRC utilisées sont à 16 ou 24 bits.

Cross-compilateur : Compilateur tournant sur une machine A et créant du code exécutable sur une machine B, d'un autre modèle. Cette technique est surtout utilisée pour le développement de programmes sur microsystèmes.

CSMA/CA : *Carrier Sense Multiple Access / Collision Avoidance*. Technique d'accès à un câble pour la réalisation d'un réseau local consistant à écouter le câble pour voir s'il est libre, puis d'émettre tout en contrôlant qu'il n'y a pas plusieurs émetteurs simultanés (cas d'une collision). Lorsqu'il y a collision, les deux émetteurs arrêtent la transmission, puis brouillent l'autre pour être sûr qu'il l'a détecté, puis attendent un temps aléatoire et recommencent le processus. Pour éviter les collisions (collision avoidance), l'émetteur envoie un train de bits qui précède le message (fonction de type "pare-choc").

CSMA/CD : *Carrier Sense Multiple Access / Collision Detection*. Même technique que CSMA/CA, mais sans train de bits précédent les messages. Technique utilisée en éthernet.

Datagramme : Un datagramme est un message circulant sur un réseau (local ou non) et qui contient toute l'information nécessaire quant à son acheminement (source, destination). Il forme un tout en tant que tel et n'a pas besoin de circuit. Une communication basée sur des datagramme doit prévoir, dans les couches supérieures, les mécanismes pour contrôler l'intégrité (séquencement, complétude des messages), vu que chaque datagramme est complet.

Diagrammes syntaxiques : Diagrammes définissant de manière non équivoque la syntaxe d'un langage. Le Pascal, par exemple, est défini au moyen de diagrammes syntaxiques.

Direct : se dit d'un adressage où le champ opérande de l'instruction contient directement l'adresse effective. Si l'on parle d'accès à un disque ou à un fichier, cela signifie que l'accès à un enregistrement donné a lieu directement, c'est-à-dire sans lecture des enregistrements précédents.

Disque : Support d'information ayant l'aspect d'une pile de 33 tours. Il s'agit d'un support magnétique, permettant la lecture et l'écriture. On distingue les "mass storage", gros disque d'une capacité de l'ordre de la dizaine à plusieurs centaines de méga-bytes et de temps d'accès de l'ordre de 5 à 80 milisecondes, des disques "cartouche" ou "cartridge" qui ont des capacités inférieures et des temps d'accès plus grands. Il ne faut surtout pas confondre unité de disque (disk-unit) et disk-pack, ce dernier devant être inséré dans l'unité pour autoriser les transferts.

DMA : Abréviation de "Direct Memory Access". Se dit de transfert allant directement du contrôleur du périphérique à la mémoire sans programme autre que celui initialisant le transfert ou traitant la fin de celui-ci.

DQDB : *Distributed Queued and Dual Bus*. Norme IEEE 802.6. Protocole de transmission basé sur des cellules de 53 bytes standardisé par l'IEEE et l'ISO. Permet des connexions orientées circuit virtuel ou sans connexion en mode datagramme.

DSA : *Directory System Agent*. Composant faisant partie du serveur d'annuaire distribué dans un contexte X.500. Il est chargé de répondre aux requête de l'agent utilisateur (DUA) et de renvoyer la requête, le cas échéant, à un autre DSA.

DUA : *Directory User Agent*. Agent utilisateur chargé de faire l'interface entre l'utilisateur et le DSA dans un contexte d'annuaire distribué (selon la norme X.500).

EBCDIC : Code de représentation interne d'information. Le code EBCDIC est surtout utilisé chez IBM et est devenu, par conséquent, un standard d'échange d'information au même titre que l'ASCII, surtout sur bande magnétique.

Editeur de lien : Programme servant à charger du code objet, à compléter les adressages non terminés et à rechercher en bibliothèque les sous-programmes manquants. Les autres termes utilisés sont : chargeur ou loader.

EMS : *Electronic Messaging Service*. Système de messagerie électronique.

Entrée-sortie parallèle : Mode de communication avec un périphérique où chaque bit d'une unité de transfert passe par une ligne séparée. A titre d'exemple, une imprimante connectée en parallèle aura 7 lignes pour passer du code ASCII, plus quelques lignes de contrôle.

Entrée-sortie série : Mode de communication avec un périphérique où les bits passent tous sur la même ligne, l'un après l'autre. C'est le mode utilisé très souvent pour la liaison entre terminal et ordinateur. Une transmission série sera soit synchrone, soit asynchrone.

EPROM : désigne une mémoire morte effaçable et reprogrammable. L'effacement a lieu par exposition à un rayonnement ultra-violet, la reprogrammation par application d'une tension environ cinq fois supérieure à celle de service. EPROM est l'abréviation de "Erasable Programmable Read Only Memory".

Ethernet : Standard développé en 1976 par Xerox. Intel et Digital se sont associés à ce projet en 1980. Il s'agit en fait d'un standard mondial qui définit le support (câble coaxial unique entre les différentes machines) et utilisant la technique d'accès CSMA/CD.

Le format des messages est le suivant :

- préambule de 7 bytes
- octet de début (start) 1 byte
- adresse destination : 6 bytes
- adresse source : 6 bytes
- longueur des données : 2 bytes
- données
- CRC

La taille d'un message est de l'ordre de 1500 bytes au maximum.

FDDI : *Fiber Distributed Data Interface*. Boucle double à passation de jeton utilisée à 100 Mbits par seconde utilisé pour la réalisation de LAN et/ou de WAN.

Fichier : Unité d'information se composant d'un ou plusieurs articles et formant un tout. Un fichier peut se trouver sur un support externe, tel que disque, bande magnétique, floppy, cartes, bande papier, etc. L'accès à un fichier par un programme peut être direct, séquentiel ou séquentiel indexé. A noter que certains supports ne permettent pas tous ces types d'accès.

Floppy disk : Support magnétique d'information consistant en un disque souple de mylar. L'organisation de l'information sur un floppy peut être soit séquentielle, soit semblable à celle d'un gros disque. La capacité de stockage d'un floppy est de l'ordre de 300 K bytes. Le floppy a une durée de vie limitée, car la tête de lecture-écriture touche la surface, contrairement à celles des disques, mais à l'avantage d'un coût nettement inférieur à un disque ou même à une bande. Le terme français consacré est : disquette.

Front-end-processor : Ordinateur frontal sur lequel l'unité centrale se décharge de certaines tâches telles que multiplexage, gestion de groupes de périphériques, télécommunications, etc.

Full duplex : Technique de transfert d'information bidirectionnelle simultanée : il faut donc un câblage pour l'aller des signaux et un pour le retour. Les terminaux branchés en full duplex nécessitent 4 fils sur un modem, à moins que la vitesse soit basse, auquel cas on utilise une seconde fréquence pour simuler la deuxième paire de fils.

Half duplex — Technique de transmission bidirectionnelle alternée; il suffira donc d'un canal pour l'aller et le retour, les transferts n'ayant lieu que dans un seul sens à la fois (Cf. full duplex).

Hardware : — Mot anglais signifiant "quincaillerie". Il désigne, par analogie, tout ce qui est matériel dans l'informatique : câbles, circuits électriques ou électroniques, ventilateurs, etc. Le terme français homologué est "matériel".

HDLC : — Technique de transmission série orientée bit. HDLC est l'abréviation de High level Data Link Control. La caractéristique en est une synchronisation au moyen d'un "flag", qui indique un début et une fin de trame, somme de contrôle incluse.

IA5 : — Jeux de caractères utilisé notamment en messagerie électronique. Ce jeux est à 7 bits, ce qui a pour conséquence l'absence des accents, point inadmissible dans des langues européennes. Ce point est celui qui fait dire à des profanes que la messagerie est inutilisable, quant bien même toutes les autres fonctionnalités sont opérationnelles.

Indexation : — Action consistant à additionner à l'adresse de l'instruction celle d'un registre dit d'index. L'avantage de cette méthode est un accès rapide à une position mémoire en fonction d'une partie d'adresse fixe (champ opérande) et d'une partie variable (registre d'index). Cette méthode est souvent utilisée pour référencer un élément d'un tableau.

Indirect : — Mode d'adressage consistant à aller rechercher, à l'adresse spécifiée dans le champ opérande de l'instruction, l'adresse définitive où se trouve l'information voulue.

Instruction — En langage évolué, une instruction est une phrase construite selon les règles spécifiques du langage et permettant d'obtenir un résultat donné. En langage machine ou assembleur, une instruction est une opération élémentaire que l'unité centrale peut effectuer, donc faisant partie du jeu d'instructions de l'unité centrale.

Interface : — Partie électronique servant à adapter les signaux internes de l'ordinateur à ceux nécessaires au contrôle d'un périphérique ou vice-versa.

Internet : — Réseau mondial utilisant la norme IP. L'accès au réseau Internet est réalisé en Suisse pour les universités, les écoles polytechniques fédérales et autres instituts d'enseignement par Switch.

Interpréteur : Programme lisant du code soit source, soit compilé et simulant son exécution après analyse du code. Cette analyse a lieu à chaque instruction, ce qui rend l'exécution d'un programme interprété plus lente que celle d'un programme compilé et exécuté.

Interrupt : Evénement extérieur au programme ou à l'unité centrale qui arrête l'exécution du travail en cours. Afin de pouvoir continuer ce travail par la suite, il est important de sauver toute l'information critique de ce travail. Cette méthode est surtout utilisée pour traiter des périphériques à leur vitesse maximale sans devoir attendre en boucle l'état "prêt" du périphérique.

IP : *Internetwork Protocol*. Protocole de niveau 3 selon le modèle ISO-OSI défini à l'origine dans le cadre d'ARPANET. Il s'agit du protocole utilisé par Internet.

IPX : Protocole de niveau 3 selon le modèle ISO-OSI défini par Novell pour les communications sous Netware.

ISO : *International Standardization Organization*. Organisation internationale de standardisation créée en 1946. Les organismes nationaux de plus de 90 pays en font partie. L'ISO publie des normes dans presque tous les domaines. La différence entre l'ISO et le CEN est le fait que le CEN, pour les pays membres (européens), publie des normes qui deviennent obligatoires pour les pays membres.

ISO 6937 : Jeux de caractères représenté sur 8 bits et utilisé en messagerie électronique notamment.

ISO-OSI : abréviation composée de ISO et OSI. Voir OSI.

Jeu d'instructions : Liste exhaustive des possibilités de l'unité centrale. Le jeu d'instructions peut être étendu par l'adjonction de nouvelles instructions au moyen de microcode.

Job : Terme anglais signifiant travail. Se dit en général d'une suite d'instructions données à un ordinateur et formant un tout. Exemple : compilation et exécution d'un programme (Cf batch).

LAN : *Local Area Network*. Réseau local multipoint. Le support physique utilisé pour un tel réseau peut être de la paire torsadée, du câble coaxial ou de la fibre optique. Les méthodes d'accès principales sont le CSMA/CD, le token ring, etc. Les vitesses sont en général de 4 ou 10 Mbits par seconde, avec passage progressivement à 16, 34, 100, voir 155 Mbits par seconde.

LAN Manager : Logiciel de gestion d'un réseau local développé par Microsoft, version publique de LAN SERVER d'IBM. D'autres versions plus étendues existent, proposées par 10Net, 3COM, DCA, etc. LAN Manager est également le nom du composant d'administration de LAN SERVER.

Lantastic : Logiciel de gestion d'un réseau local. Ce logiciel est particulièrement intéressant dans son rapport coût/performance pour des petits réseaux.

Langage : Moyen de donner à un ordinateur les ordres nécessaires à un traitement informatique. Un langage répond à des règles de syntaxe bien définies et strictes. Citons, à titre d'exemples, le Fortran, le Basic, le Pascal, le Cobol, le PL/1, l'Ada, l'Algol, l'APL et le Lisp. La syntaxe du langage peut être définie de plusieurs manières, dont la notation BNF ou au moyen de diagrammes syntaxiques. On distingue en général le langage assembleur, défini par l'unité centrale sur laquelle on travaille, d'un langage évolué, qui est indépendant de la machine.

LDA : *Loi sur les Droits d'Auteur*. Loi définissant ce qu'est une oeuvre et assimilant les programmes informatiques à des oeuvres. Mise en vigueur en Suisse le 1er août 1983, elle régit les droits de reproduction des logiciels notamment.

Macro : se dit d'une séquence d'instructions avec ou sans paramètre, pouvant être appelée au moyen d'une nouvelle instruction. Le macro-pré-processeur remplacera alors cette nouvelle instruction par les instructions spécifiées. Il est a noter que les macros sont traitées avant ou pendant la compilation, mais non en cours d'exécution ; c'est ce qui les différencie des sous-programmes.

MAC : *Media Access Controler*. Contrôleur d'accès à un câble réseau. Mot utilisé souvent pour parler de l'adresse MAC, c'est-à-dire l'adresse éthernet fixée dans le contrôleur et unique au monde.

MAN : *Metropolitan Area Network*. Réseau regroupant des WAN et des LAN et ayant une certaine taille, de l'ordre de grandeur de la ville.

Mémoire : La mémoire est le lieu où sont stockés le programme et ses variables. Elle est composée soit de tores magnétiques, en ferrite, soit de circuits électroniques, auquel cas on parle de mémoire "MOS". De nouveaux types de mémoires sont apparus sur le marché dernièrement, telles les mémoires à bulles. La mémoire d'un ordinateur est caractérisée par le fait que le "CPU" y a accès de manière directe et aléatoire, en un temps aussi bref que possible (de 80 nanosecondes à 2 microsecondes environ). C'est en mémoire que le "CPU" cherche les instructions qu'il doit exécuter, ainsi que les opérandes de ces instructions. Certains périphériques utilisés pour de gros transferts peuvent avoir un contrôleur permettant l'accès direct à la mémoire ; c'est surtout le cas des disques.

Microcode : Le microcode est la définition d'une instruction qui ne peut être exécutée telle quelle par l'unité centrale. Il est contenu dans des ROMs ou RAMs (voir ces mots) à accès très rapide et définit les opérations au niveau le plus bas que l'unité centrale doit effectuer. (exemple : ouvrir telle porte, passer l'information à tel additionneur, etc.)

Microsoft : Entreprise américaine leader du marché en ce qui concerne toute une gamme de logiciels.

Mnémonique : Terme désignant une abréviation d'un code d'opération. Ainsi, par exemple, l'instruction "load A register with contents of ..." pourra se dire "LDA ...". Le langage assembleur utilise surtout des mnémoniques.

Modem : *MOdulateur DEModulateur* : appareil servant à moduler sur une fréquence les signaux destinés à un périphérique se trouvant à distance. Le modem permet de travailler avec un terminal à distance du système en utilisant soit une ligne louée, soit une ligne commutée du réseau téléphonique.

Mot : Unité de base selon laquelle est construit un ordinateur. Les microprocesseurs sont souvent basés sur des bytes (un mot = un byte), alors que les mini-ordinateurs sont plus souvent basés sur des mots de 16 ou 32 bits. Certains grands ordinateurs utilisent des mots de 36 bits (exemple : U1160, HB66/20P), de 48 bits (CDC 3600-3800), de 60 ou même 64 bits (CDC 6000). La taille du mot est le nombre de bits répondant à une adresse mémoire donnée. C'est la taille du mot qui définira la précision à laquelle travaillera l'unité centrale.

MS : *Message Store*. Entité de sauvegarde, dans un contexte de messagerie électronique, des messages.

MS Mail : Système de messagerie de Microsoft. Il s'agit d'un système de messagerie propriétaire auquel on peut adjoindre une passerelle permettant d'atteindre le monde X.400. En 1994, environ un millier de postes utilisateur sont équipés de MS Mail à l'Etat de Genève.

MTA : *Message Transfert Agent*. Entité de transfert des messages dans un contexte de messagerie électronique selon la norme X.400. D'autres messageries, telles que MS Mail, utilisent le terme de bureau de poste.

Multiplexeur : Appareil permettant de connecter plusieurs terminaux à un ordinateur en n'utilisant qu'une ligne de télécommunication. En général, l'entrée-sortie des terminaux sera asynchrone, tandis que du côté ligne, on travaillera en synchrone.

Multiprocessing : Technique consistant à avoir plusieurs tâches en cours d'exécution, parfois dépendantes, parfois indépendantes (Cf temps réel).

Multiprogrammation : Technique consistant à exécuter plusieurs travaux de types divers (compilation, exploitation, mise au point, etc.) en même temps et indépendamment les uns des autres. La multiprogrammation permet de laisser dans un état "dormant" un travail en attente d'un périphérique, par exemple, tout en exécutant un autre travail. Le time-sharing est une application sophistiquée de multiprogrammation.

NETBEUI : Nom d'origine du protocole pour l'implémentation de NETBIOS par Microsoft pour IBM. A noter que NETBEUI n'est pas routable.

NETBIOS : *NETwork Basic Input Output System*. Interface de programmation défini par IBM et Microsoft permettant la communication dans un contexte LAN. Cet interface permet tant l'utilisation de circuits de communication que l'envoi de datagrammes. Il ne défini par contre pas le support de communication même, qui peut être de type NETBEUI ou RFC 1006 (utilisant le protocole IP pour communiquer.

Netware : Logiciel de gestion de réseau local de micro-ordinateurs développé par Novell, fabriquant américain de logiciel.

NT : *New Technology*. Nouveau système d'exploitation de Microsoft qui a pour caractéristique principale de garder Windows comme base, mais en supprimant le DOS comme couche inférieure.

Objet : Résultat provenant d'une compilation. Le code objet contient les instructions telles qu'elles devront être chargées, mais il manque encore les adresses définitives, qui seront déterminées précisément lors de l'édition de liens. Les bibliothèques de sous-programmes sont en général à disposition de l'utilisateur sous forme de code objet. On appelle parfois un fichier contenant du code objet "relogeable".

Octet : Ensemble de 8 bits considérés comme un tout. On utilise en général l'octet comme unité en parlant de capacités. En utilisant le code ASCII, on stocke un caractère dans un octet. (le terme anglais consacré est "byte").

OSF *Open Software Fundation*. Association crée en 1988 par plusieurs constructeurs et fournisseurs de logiciels pour définir un environnement applicatif commun. OSF et Unix ont des buts identiques et sont donc en concurrence.

Paging : Technique de gestion de l'espace mémoire physique dans un contexte temps réel ou time-sharing. Chaque tâche dispose sur disque d'une image de son espace mémoire, divisé en pages (généralement de 1 K mots ou plus). Lorsqu'un programme est exécuté, il fera charger par le système d'exploitation la ou les pages nécessaires. Cette technique nécessite, du point de vue hardware, une table de conversion des adresses utilisées par les programmes en adresses physiques, et du point de vue software un module de gestion des demandes de chargement en mémoire des pages non résidentes. La taille mémoire disponible virtuellement peut donc être supérieure à la taille mémoire réelle. On appelle aussi cette technique : mémoire virtuelle.

Paramètre : Le paramètre est la partie qui varie lors d'un appel à un sous-programme et qui n'est pas nécessairement la même variable à chaque appel. Le chapitre concernant le passage des paramètres du programme appelant au sous-programme serait long; on ne citera donc que quelques méthodes : par adresse : on passe au sous-programme l'adresse des paramètres (cas du Fortran et du Pascal (var:)). par nom : on fait la concordance entre le nom des variables. Méthode très rare. par pile : on pose les paramètres sur la pile principale, à l'appel comme au retour. Cas utilisé par le Forth. par registre : les valeurs à transmettre sont stockées dans les registres de l'unité centrale. Méthode très limitée, le nombre de registre d'une machine étant fini et petit (ne s'utilise en général qu'en assembleur). par valeur : on copie les paramètres dans une zone mémoire tampon qui est propre au sous-programme (cas du Pascal).

Parité : Bit(s) ajouté(s) à l'information pour en vérifier l'intégrité. Lorsque l'on parle de caractères, la parité peut être paire (toujours un nombre pair de bits positionnés à 1) ou impaire, ou forcée à un, ou nulle. La vérification de la parité peut être utile lorsque l'on transmet de l'information à haute vitesse ou sur des lignes de qualité médiocre.

Périphériques : Eléments ou machines connectées à l'ordinateur permettant les communications avec l'extérieur. Il peut s'agir de terminaux, de lecteurs de cartes ou de bandes perforées, de perforateurs de cartes ou bandes, d'imprimantes, d'unités de bandes magnétiques ou de disques, etc. On distingue les périphériques lents, qui permettent un transfert caractère par caractère, avec délais possibles entre eux, des périphériques rapides où le transfert à lieu par blocs d'information (disques, bandes).

PIN : *Personnal Identification Number*. Numéro personnel d'identification, souvent utilisé dans les administrations pour identifier les administrés. Le PIN est en contraction, partielle du moins, avec le respect de la sphère privée.

Plotter : Périphérique pouvant tracer des courbes. L'ordinateur envoie au plotter soit des coordonnées, soit une direction et une distance et l'ordre de tracer ou de déplacer la plume sans tracer.

PRMD : *Private Management Domain*. Composé d'un ou plusieurs MTA, le PRMD représente un domaine privé de gestion de messagerie électronique.

PROM : Abréviation de "Programmable Read Only Memory". Il s'agit de mémoire morte que l'on peut programmer une seule fois. (Cf aussi EPROM)

Protocole de télécommunication : Règles, très strictes, définissant les questions et les réponses devant avoir lieu lorsque deux équipements informatiques sont en communication. Ces règles prévoient des procédures de récupération en cas d'erreur de transmission ou de "time-out" (réponse non parvenue dans les délais).

Public Data Network : Nom anglais de "réseaux public de transmission de données". Il s'agit d'un réseau analogue à celui des téléphones, mais réservé aux transmissions de données. Peuvent donc se connecter sur ce réseau principalement les ordinateurs et les terminaux, mais également certains autres appareils utilisant des techniques digitales telles que télécopieurs, télex, etc. Le réseau suisse de transmission de données, qui devra être opérationnel vers fin 1982, se nomme TELEPAC et, au départ, sera composé de trois centraux, l'un à Berne, un autre à Genève et le troisième à Zurich. Il ne sera connecté au réseau européen, EURONET, qu'une ou deux années plus tard. Les autres réseaux nationaux dont on entend parler sont principalement TRANSPAC (France), TYMNET et TELENET (USA), DATAPAC (Canada), DATEX-P (Allemagne) et DDX-2 (Japon). Ils travaillent tous en utilisant la norme X.25 du CCITT, avec, toutefois, quelques variantes. Le concept de ces réseaux a été développé dans les milieux universitaires aux Etats-Unis et la mise en service a eu lieu sous le nom d'ARPANET dès 1969.

RAM : Mémoire vive fondée en général sur la technologie MOS. RAM signifie "Random Access Memory".

Réentrance : Méthode de programmation permettant à plusieurs programmes d'utiliser un code commun de manière indépendante, donc sans vérification si le passage précédent est terminé. Il est donc important de bien gérer les variables et les registres afin qu'il n'y ait pas d'interaction d'un programme sur un autre, malgré la partie commune.

Registre : Elément de l'unité centrale où l'on stocke de manière temporaire des valeurs utilisées comme opérandes par les instructions du programme. Un registre permet l'accès, à très grande vitesse, à une valeur. On distingue en général les registres arithmétiques (ou registres principaux) des registres d'index, d'adresse ou d'état de la machine.

Relogeable : se dit du code objet lorsque les adresses ne sont pas encore définitives, mais seulement exprimées comme déplacement par rapport à une base. Les bibliothèques de programmes et les résultats de la plupart des compilations sont en général en code objet relogeable.

Remote Job Entry : Station de soumission de travaux batch à distance. Il s'agit généralement d'un terminal lourd composé au moins d'un lecteur de cartes ou d'un autre périphérique de lecture et d'une imprimante.

Répéteur : Appareil permettant de relier un ou plusieurs réseaux locaux en faisant de l'amplification des signaux uniquement. Le répéteur n'a aucune fonction de tri des paquets circulant. La norme éthernet permet le passage par quatre répéteurs au maximum pour transmettre un message d'un poste quelconque à un autre.

Réseau : On parle de réseau lorsque plusieurs ordinateurs sont interconnectés et permettent le travail ou les transferts de données de l'un à l'autre. On distingue en général les réseaux publics (PDN) et les réseaux locaux (à l'intérieur d'un bâtiment ou d'une entreprise, par exemple). Les réseaux peuvent être construit en étoile autour d'un ou plusieurs centraux (cas des réseaux publics), en cercle ou sur une ligne en général coaxiale. Le premier grand réseau mis en service dès 1969, a été ARPANET, qui est considéré comme l'ancêtre des réseaux actuels.

RFC *Request For Comment*. Procédure utilisée surtout dans le monde Internet pour la définition de normes ou d'explication de fonctionnement de certains logiciels. Les RFC sont numérotés et sont des normes de fait, souvent référencés par des constructeurs dans leur spécifications logicielles.

ROM : Mémoire morte dont le contenu est défini à l'usine. Elle n'est pas reprogrammable, contrairement aux EPROM. L'avantage par rapport aux EPROM est la rapidité d'accès par rapport au prix.

Routeur : Le routeur est un appareil qui permet de filtrer et transmettre d'un réseau à un autre des paquets en se basant sur des adresses réseau et/ou des protocoles de transmission. Le routeur travaille au niveau 3 du modèle ISO-OSI.

Séquentiel : Accès à une information stockée en général sur un support externe dans l'ordre où elle se trouve sur ce support externe. Il faudra donc lire tous les enregistrements précédant celui que l'on recherche.

Séquentiel indexé : Méthode d'accès à un fichier permettant de retrouver un article en fonction d'une clé d'accès définie au moment où la structure du fichier a été créée. Cette clé permet au système un accès direct à l'information voulue.

Signalisation : La signalisation se rapporte aux informations circulant dans un réseau pour l'établissement de la liaison entre l'appelant et l'appelé, pour la facturation et la maintenance du réseau.

Simplex : Technique de transmission unidirectionnelle. A ne pas confondre avec half duplex, qui est bidirectionnel (simplex alterné).

Software : Partie de l'ordinateur qui n'est pas le matériel. Ce terme est utilisé par opposition à "Hardware". Le software comprend en général tous les programmes permettant une saine exploitation d'un ordinateur, les sous-programmes de bibliothèque, les compilateurs, les interpréteurs, chargeurs, etc. Le terme homologué en français est "Logiciel".

SONET : *Synchronous Optical Network*. Standard établi par l'ANSI et adopté par le CCITT sous le nom de SDH permettant l'interconnexion au niveau 1 du modèle ISO-OSI et travaillant à des vitesses de 51.84 Mbits par seconde à 9,9 Gbits par seconde.

Sous-programme : Morceau de programme séparé du programme principal et pouvant être appelé de plusieurs endroits. Un sous-programme doit toujours connaître l'adresse de retour à laquelle il transférera le contrôle une fois qu'il sera terminé.

Spooling : Action consistant à simuler sur disque des entrées-sorties, qui ralentiraient un programme, et de faire les transferts réels en différé. Cette technique est surtout utilisée pour la gestion des imprimantes.

Swapping : Action consistant à transférer sur disque une zone mémoire complète en vue de la libérer pour une autre tâche. La suite du travail aura lieu une fois cette zone mémoire rechargée.

Synchrone : Transmission série entre terminaux et/ou ordinateurs ayant lieu par blocs de plusieurs caractères. Chaque bloc est précédé par deux à huit caractères de synchronisation et suivi par zéro à deux caractères de fin. L'avantage des transmissions synchrones réside en un taux d'efficacité plus grand (moins de bits perdus) sur des transmissions relativement importantes.

TCP : *Transmission Control Protocol*. Protocole de transport orienté byte, équivalent à une classe de transport 4 de l'ISO.

Télétraitement : Technique d'utilisation des ordinateurs à distance au moyen de terminaux reliés au site central par lignes téléphoniques, télégraphiques ou hertziennes.

Temps réel : Mode d'exploitation d'un ordinateur dans lequel plusieurs programmes peuvent travailler simultanément. Chaque programme a une priorité donnée et ne pourra prendre le contrôle de la machine que si aucun programme de priorité supérieure ne doit tourner. En général, ce sont des événements extérieurs qui déterminent si un programme doit attendre ou faire quelque chose. Un programme ayant pris le contrôle pourra, en principe, le garder jusqu'à ce qu'il ait terminé le travail en cours ou qu'un événement prioritaire ait eu lieu (interruption).

Terminal : Unité périphérique se composant généralement d'un clavier et d'un moyen d'affichage. Il peut être situé à distance de l'ordinateur même. Les terminaux peuvent se diviser en deux grandes catégories : ceux avec support papier (ASR, KSR, etc) et ceux sans support papier (écrans).

Timer : Circuit logique permettant une mesure du temps indépendamment du programme.

Time-sharing : Mode d'exploitation d'un ordinateur consistant à répartir le temps entre utilisateurs par petits morceaux appelés "time-slice". Il s'agit donc de partager, dans le temps, les ressources d'un ordinateur de telle manière que chacun ait plus ou moins l'impression d'être le seul utilisateur.

Time-slice : Tranche de temps accordée à un utilisateur dans un contexte "time-sharing".

UA : *User Agent*. Agent utilisateur, c'est-à-dire interface homme machine en général, qui sert à communiquer avec le MTA dans un contexte de messagerie électronique, X.400 ou non.

UIT : *Union Internationale des Télécommunications*. Emanation des administrations postales au niveau international, l'UIT regroupe différentes entités, dont le CCITT.

Unix : Système d'exploitation d'ordinateurs que l'on retrouve sur des plate-formes très variées, propriétaires ou basées sur la gamme Intel ou Motorola.

WAN : *Wide Area Network*. Si le LAN est purement local, typiquement localisé à un bâtiment, le WAN permet l'interconnexion de plusieurs LAN pour en former un réseau à l'échelle d'une entreprise par exemple.

X.25 : Norme de télécommunication édictée par le CCITT (comité consultatif international téléphonique et télégraphique) sur l'instigation des constructeurs. Cette norme est utilisée de plus en plus et est celle utilisée par les réseaux publics tels que Euronet, Telepac, Transpac, Tymnet, Telenet et Datapac.

X.400 : Recommandation du CCITT en matière de messagerie. Elle regroupe en fait toutes les normes de la gamme des X.400 pour définir clairement l'interconnexion au niveau des MTA et des UA. Le système d'adressage est composé de : un pays (Country), d'une administration (ADMD), d'un domaine privé (PRMD), d'une organisation (Organization), d'une à quatre unités d'organisation (Organizational Units), d'un nom (Surname), d'un prénom (given name). Quelques-uns de ces champs sont facultatifs. D'autres champs, tels que senior ou junior, numéro de poste, etc. existent.

X.500 : Recommandation du CCITT qui défini un système de gestion d'annuaire décentralisé et utilisable au niveau mondial. X.500 est souvent considéré comme l'annuaire de la messagerie X.400, mais en fait cette norme a un domaine d'application beaucoup plus large et dépassant de loin le simple annuaire de messagerie. X.500 n'a aucun lien avec X.400.

23.- Bibliographie

Il n'est pas possible de citer une bibliographie exhaustive dans un domaine où la progression des connaissances est très rapide. Nous ne citerons ici que quelques revues et manuels, sans détails particuliers, auxquels le lecteur pourra se reporter pour des compléments d'information. Ce sont ces mêmes sources qui nous ont permis l'élaboration de ce fascicule, et, en général, la sources exacte n'est pas précisée.

Revues périodiques

- Byte, Mac Graw Hill, P.O. Box 328, Hancock, New Hampshire 03449,
- Communications of the ACM, publication de l'"Association for Computer Machinery", P.O. Boc 12114, Church Street Station, New York,
- Kilobaub microcomputing, Microcomputing, P.O. Box 997, Farmyngdale New York 11737,
- L'ordinateur individuel, 27 rte du Grand Mont, 1052 Le Mont sur Lausanne,
- Microsystèmes, 2 rue de Bellevue, 75940 Paris,
- Temps réel, Compagnie française d'information pour les entreprises, 40 rue du Colisée, 75381 Parix Cedex 08.

Manuels

- Synchronisation de programmes parallèles : expression et mise en oeuvre dans les systèmes centralisés ou distribués, Françoise André, ISBN 2-04-015606-2.
- Computer operating systems, D. W. Barron, Chapmann and Hill, 1973, ISBN 0-07-017603-5.
- La bureautique, outils et applications, Jean-Paud de Blasis, Editions d'organisation, ISBN 2-7081-0489-6.
- Operating System Principles, Per Brinch Hansen, Prentice-Hall, Inc, 1973, ISBN 0-13-637843-9.
- The architecture of Concurrent Programs, Brinch Hansen, Prentice-Hall, Inc., 1977.
- Contrôle de l'accès aux objets partagés dans les systèmes informatiques, Monographies d'informatique de l'AFCET, ISSN 0181-9283.
- Systèmes d'exploitation des ordinateurs, Crocus, Dunod, 1976.
- Communication architecture for distributed systems, ISBN 0-201-14458-1.
- Computer Networks and their protocols, D. W. Davies, Wyley, ISBN 0-471-99750-1.

- An introduction to operating systems (Unix, Vax, CP/M, MVS, VM), Harwey-M. Deitel, ISBN 0-201-14502-2.
- Réseaux téléinformatique, J.-C. Derniame et alias, Hachette technique, ISBN 2-01-006219-1.
- The structure of the T.H.E. multiprogramming system, Dijkstra E. W., Comm. ACM, vol. 11, no 5, May 1968, p 341-346.
- System programming, John J. Donovan, Mac Graw Hill, ISBN 0-07-017603-5.
- Structured Concurrent Programming with Operating Systems Applications, Holt, R. C., Lazowska, E. D., Graham, G. S., Scott, M. A., Addison-Wesley Publishing Company, 1978.
- Compiler design theory, Philip-M. Lewis, ISBN 0-201-14455-7.
- Fundamental of Operating Systems, Lister, A. M., Springer-Verlag, 1979.
- Operating systems, Harold Lorin, ISBN 0-201-14464-6.
- Gestion de fichiers et bases de données, N. Magnenat-Thalmann, Gaëtan Morin, ISBN 2-89105-042-8.
- Technical aspect of telecommunication, John E. McNamara, Digital, ISBN 0-93376-01-0.
- Principes de la commutation par paquet, Joseph Pitteloud, tiré à part de la revue Output 9/1980 à 5/1981, 9403 Goldach.
- Structured computer organization, par Andrew S. Tannenbaum, Prentice-Hall, ISBN 0-13-854505-7.
- Conception et implantation de langages de programmation, D. Thalmann et B. Levrat, Gaëtan et Morin, ISBN 0-88612-020-9.
- Local computer network technologies, Carl Tropper, Academic Press, 1981, ISBN 0-12-700850-0.
- Algorithms + data structure = programs, Niklaus Wirth, Prentice Hall, ISBN 0-13-022418-9.
- Les virus, méthode et techniques de sécurité, Jean-Claude Hoff, Dunod, 1991.

Brochures et textes divers

- Représentation interne de l'information, D. Petitpierre, Some-Bits 11, publication du CCEES, Genève, juin 1980.
- Algorithmes et classification des langages, D. Thalmann, Some-Bit 55, publication du CCEES, Genève, 1976.
- Structure de données, D. Thalmann, Some-Bits 56, publication du CCEES, Genève, 1976.
- La compilation, D. Thalmann, Some-Bits 58, publication du CCEES, Genève, 1976.
- Petit Glossaire de Termes Informatiques (PGTI), Some-Bits 60, quatrième édition, publication du CCEES, Genève, mai 1983.
- Glossaire de termes informatiques réalisés par Jérôme Pönitz, Centre Cantonal d'Informatique, 26 rue du Stand, 1205 Genève.

- Eléments d'un réseau Ethernet IEEE 802.3 et ISO 8802.3, par Jean-Pierre Gilliéron, Département des travaux publics et de l'énergie, service des télécommunications et des installations informatiques, 1990.
- Introduction à X.500, par Jean-François Paccini, CUI, Université de Genève, 1993.

Notes de cours

- Cours "Architecture des systèmes time-sharing", Prof. C. A. Héritier, Université de Genève.
- Cours "Connaissance des ordinateurs", Prof. P. Zanella, Université de Genève.
- Cours "Connaissance des ordinateurs", Jean-Bernard Roux, ESC Malagnou, Genève.
- Cours "Connaissance des ordinateurs", Gérard Ineichen, Collège Calvin, Genève.
- Cours "Technique de recherche d'erreurs", Gérard Ineichen, ESC Malagnou, Genève.
- Cours "Systèmes d'exploitation", A. Schiper, EPFL Lausanne, 1982.

Index alphabétique