SYSTEMES D'EXPLOITATION

1. NAISSANCE DE L'INFORMATIQUE

L'**Informatique** (des mots « information » et « automatique ») est la science du traitement automatique et rationnel de l'information. On entend par traitement: la manipulation des informations (tri, recherche, modification, …) ou le calcul plus ou moins compliqué permettant d'obtenir un résultat (résolution d'une équation ou d'un problème numérique). L'informatique est considérée comme le support des connaissances et des communications dans les domaines scientifiques, économiques et sociaux.

Les besoins de l'informatique sont nombreux. Elle est utilisée pour la gestion des petits ou grands volumes d'information et leurs échanges de données (gestion des entreprises, banques, bases de données, minitel…), la création artistique (texte, musique, dessin…), le traitement des images (reconstruction, correction des déformations, amélioration des contrastes, reconnaissance de formes…), la médecine (tomographie, scanner, gestion hospitalière, recherche…), la conception d'objets ou de produits (industrie automobile ou aéronautique, calculs des formes et des structures, résistances aux déformations et aux chocs, aérodynamique…), le suivi de la production, la simulation d'expériences, etc.

Un **Ordinateur** (dont le mot suggère l'ordre logique) est une machine programmable. Il permet le traitement des informations numériques ou discrètes par opposition aux informations analogiques ou continues telles que les grandeurs physiques classiques: positions, intensités de courant, etc.

La manipulation de l'information discrète remonte à la plus haute antiquité. Pour effectuer des calculs ou mémoriser des résultats, on utilisait des cailloux, des bouliers ou encore d'autres objets.

C'est l'écossais John Napier (1550-1617) qui a effectué la transition entre le boulier et les premières machines à calculer qui sont alors mécaniques. Sa machine (ABAQUE) donnait le résultat d'une multiplication grâce à une série de cubes emboîtés les uns dans les autres et formant plusieurs colonnes.

Au **XV^{e}-XVI^{e} siècles**, l'invention de l'échappement des horloges introduisit un découpage du temps en intervalles égaux permettant de le mesurer. L'invention des automates à séquences fut une étape fondamentale dans l'informatique. A cette époque, les programmes qui effectuaient l'animation des automates étaient totalement intégrés aux machines. Ils étaient figés mécaniquement et donc pour changer de scénario il était nécessaire à chaque fois de recréer une machine. De tels automates existent encore de nos jours: les *jacquemarts* de l'horloges de la Cathédrale de Strasbourg (1554) en sont un exemple.

Au **XVII^{e} siècle**, apparaît la première machine à calculer: l'*Horloge Calculante* qui effectuait des additions et des soustractions pour des calculs en astronomie. Inventée en 1623 par Wilhem Schickard (1592-1635), elle fonctionnait à l'aide de roues dentées et de repères mémorisant les résultats intermédiaires. Une clochette retentissait lorsqu'un dépassement de capacité de la machine survenait. La machine fut détruite après la mort de Schickard, seul un plan fut conservé.

Blaise Pascal (1623-1662) réalisa une machine arithmétique (1642) pour aider son père, collecteur d'impôts, à effectuer des additions, des soustractions et des conversions (de monnaies). Il s'agissait d'une machine à programme figé qui utilisait des roues dentées à dix positions. Les données ne faisaient pas partie de la machine, elles étaient entrées à chaque utilisation. Cette approche donna un caractère déjà plus universel à cette calculatrice. Une dizaine d'exemplaires de ces machines existent encore.

Au **XVIII^{e} siècle**, le philosophe et mathématicien allemand Gottfried Wilhem von Leibniz (1646-1716) construisit une machine effectuant quatre opérations: l'addition, la soustraction, la multiplication et la division (roue de Leibniz). De son côté, le grenoblois Jacques De Vaucanson (1709-1782) mit au point de nombreuses machines: pompe à élever les eaux, perceuse, premier métier à tisser, le joueur de flûte traversière (1737), le canard (1738), le joueur de tambourin (1738), etc. Vers la fin du siècle, Joseph-Marie Jacquard (1752-1834) inventa un métier à tisser, qui au lieu d'avoir un programme figé comme dans les Jacquemarts, possédait un programme à l'extérieur sur des cartes perforées (plus besoin de créer une nouvelle machine pour produire un motif de tissage différent). On retrouve ce principe de machines à programmes extérieurs dans les limonaires, les boîtes à musique et les pianos mécaniques du siècle dernier.

Les premiers développements importants pour le traitement de l'information ont lieu à la fin du **XIX^{e} siècle**. Les dispositifs qui étaient alors mécaniques deviennent électromécaniques (à cames et électroaimants), puis électriques (à relais) et enfin électroniques.

Sur le modèle de la machine de Leibniz, le Français Charles X. Thomas (1785-1870) construisit l'arithmomètre (1820), machine portable et pratique qui fut vendue à plus de quinze mille exemplaires en trente ans. C'est en 1822 que Charles Babbage (...-1871) construisit la machine différentielle. Non complètement satisfait de sa première réalisation, il chercha à l'améliorer jusqu'en 1833. Pendant cette période un certain nombre de techniques d'ingénierie furent développées. Après l'abandon de la machine différentielle, Charles Babbage mit au point sur le papier la machine analytique (qui ne vit jamais le jour). Il ajouta l'utilisation de mémoire pour les résultats intermédiaires et la possibilité de choix dynamique et automatique des instructions à exécuter.

Bien que l'utilisation de cartes perforées soit antérieure à 1725 soit avant même la révolution industrielle, ce n'est qu'en **1890**, que **H. Hollerith** eut l'idée d'utiliser une machine de tri avec des cartes perforées pour effectuer le recensement des Américains.

Un idée très importante, s'appuyant sur les travaux de **Georges Boole** (1815-1864) en **1848** (l'algèbre de Boole) et sur la thèse de **Claude Elwood Shannon** (1916-) en **1938** (qui par la suite fonda la théorie de l'information) fut l'utilisation de la représentation des informations sous forme binaire, c'est à dire sous 2 états: vrai ou faux, présent ou absent, 0 ou 1.

Durant la seconde guerre mondiale, des calculateurs puissants furent créés pour effectuer des calculs de balistique ou permettre le décodage des messages cryptés.

En 1939, le prototype du premier calculateur vraiment numérique fut réalisé à l'Iowa State College. Baptisé "ABC", il était la première machine à utiliser des tubes à vide comme circuits logiques. Les deux concepteurs John Atanasoff et Clifford Berry donnèrent leurs noms à cette machine : "Atanasoff Berry Computer".

En **1943**, Colossus le premier calculateur électronique fut construit à des fins de décryptage des messages générés par **Enigma** la machine de chiffrement allemande. **Alan Turing** faisait partie des concepteurs. Colossus comportait 5 processeurs pouvant traiter chacun 5000 caractères par seconde. Grâce à des registres spéciaux et à une horloge interne, les processeurs pouvaient travailler en parallèle, ce qui donnait à Colossus une vitesse de 25000 caractères par seconde! Cette vitesse élevée fut capitale dans les efforts de décryptage mis en oeuvre pendant la dernière guerre mondiale.

Le premier calculateur électromécanique commandé par programme, l'Automatic Sequence Controlled Calculator (**ASCC**) Mark I, a été crée en 1944 par **Howard Hataway Aiken** qui s'est inspiré des plans de la machine analytique de Charles babbage datant

d'une centaine d'années plus tôt. Cette machine pouvait additionner, soustraire, multiplier, diviser, calculer des puissances et des logarithmes de 10, effectuer des calculs trigonométriques de type sinus et cosinus. Il fallait 3 secondes au calculateur pour effectuer une multiplication. Une bande de carton perforée portait les instructions et des relais téléphoniques contrôlaient des registres à roues numérotées. Le mark I avait une longueur de 15 mètres et une hauteur de 2,40 mètres pour pratiquement 800 km de câblage. Ce calculateur fut par la suite utilisé à l'Université de Harvard pendant près de 15 ans. Le premier supercalculateur Cray 1S (1981) avait 10 km de câbles et son concurrent le Cyber 205 de Control Data Corporation en avait 700.

En 1946, le premier calculateur numérique de grande puissance entra en fonctionnement. L'**ENIAC** (Electronic Numerical Integrator and Calculator) était programmé au moyen d'un système d'interrupteurs et de fiches montés extérieurement. Il fut construit par **J.Presper Eckert, John Mauckly** et **Goldstine**. Constitué de milliers de tubes, cette machine, qui resta opérationnelle jusqu'en 1956, disposait de branchements conditionnels et était programmable manuellement. Elle possédait une faible capacité de mémorisation. Une addition s'effectuait en 200 ms.

Il est admis que la naissance de l'informatique se situe en **1946**, date à laquelle **Johannes Von Neumann** (1903-1957) énonça le principe du premier ordinateur:

1) Possibilité de mémorisation des résultats partiels étendue à l'enregistrement des programmes et des données,
2) Exécution séquentielle des instructions enregistrées avec possibilité de branchement conditionnel.

Ce sont ces caractères d'adaptabilité aux problèmes posés (changement de programmes ou appel de sous-programmes spécifiques) qui rendirent la machine de Von Neumann universelle. C'est aussi ce qui la différenciait des calculateurs précédents.

Plus précisément, un ordinateur doit disposer en fait des fonctions de base suivantes:

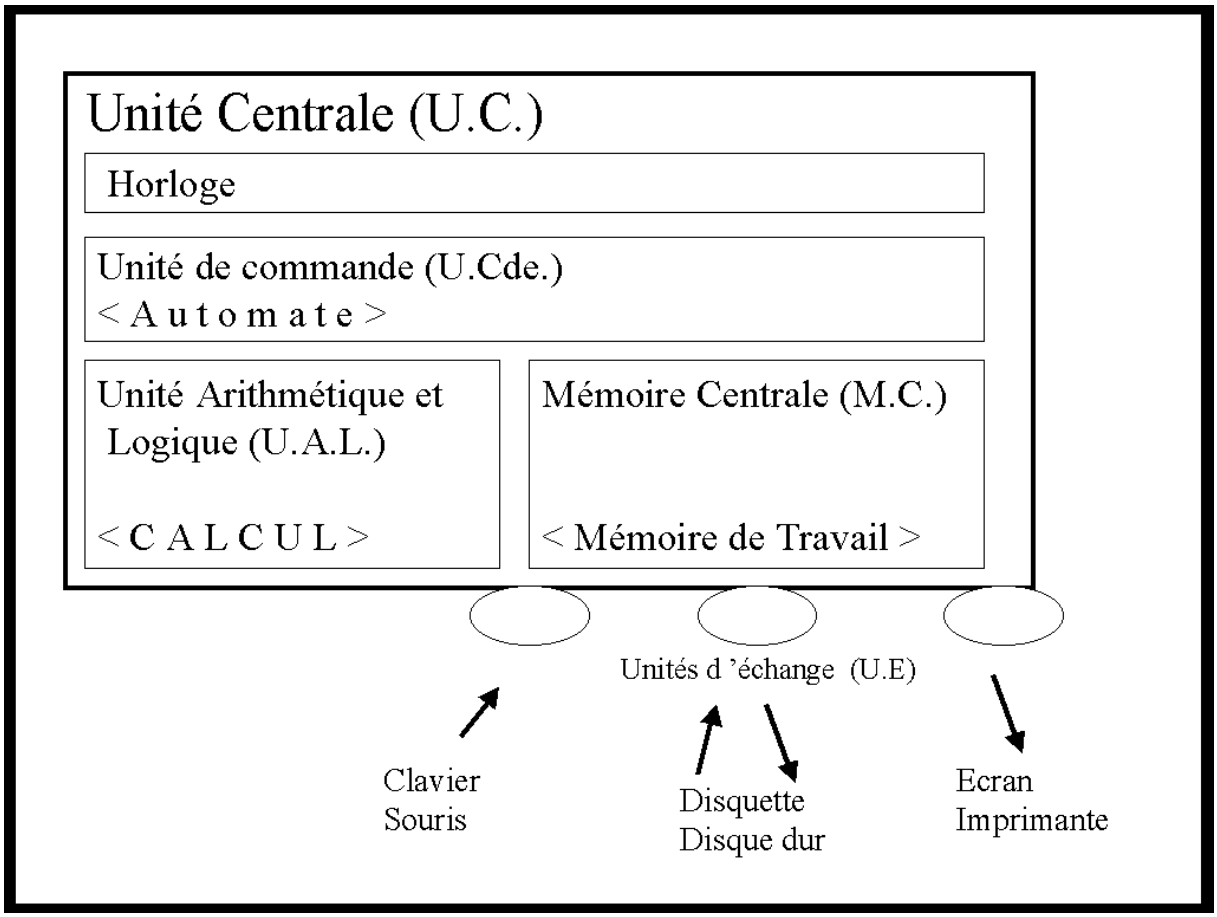
1) moyen d'entrée (échange et recensement des informations),
2) moyen de mémorisation des informations,
3) moyen de calcul (traitement de l'information),
4) moyen de sortie (résultats),
5) moyen de décision (choix entre des éventualités, l'une d'entre elles en fonction de résultats antérieurs),
6) gestion des données et des instructions (stockées dans le moyen de mémorisation sous la même forme et accessibles de la même façon).

Les idées 1, 2 et 4 ont été introduites dès 1822 par le mathématicien Charles BABBAGE. Celui-ci a rajouté les idées 3 et 5 vers 1830. L'idée 6 a été introduite par Von Neumann en 1947 (machines à programmes enregistrés dites machines de Von Neumann). Dans la machine de Von Neumann, on peut distinguer deux composantes fondamentales:

1) l'unité centrale qui permet la mémorisation et le traitement des programmes et des données,
2) l'unité d'échange pour les communications avec l'extérieur.

De nos jours, on peut schématiser un ordinateur par une Unité Centrale (de traitement de l'anglais « Central Processing Unit ») dans laquelle on distingue:

- l'**horloge** qui permet le cadencement des opérations élémentaires,
- la **Mémoire Centrale** (ou mémoire de travail) qui reçoit temporairement les programmes et les données,
- l'**Unité Arithmétique et Logique** qui effectue les traitements sur les données et
- l'**Unité de Commande** (appelée encore Unité de Contrôle) qui interprète les instructions machines et orchestre le transfert des informations à travers des voies de connexions électroniques appelées BUS.



Le mot processeur évoque de nos jours le micro processeur. Une définition plus juste serait « tout moyen matériel ou logiciel pour exécuter un processus ».

DE L'ELECTROMECANIQUE A L'ELECTRONIQUE.

En 1947, John Bardeen, Walter H.Brattain et William Shocley inventèrent aux laboratoires Bell le premier transistor (transfer resistor = résistance de transfert). Ces inventeurs reçurent le prix Nobel en 1956.

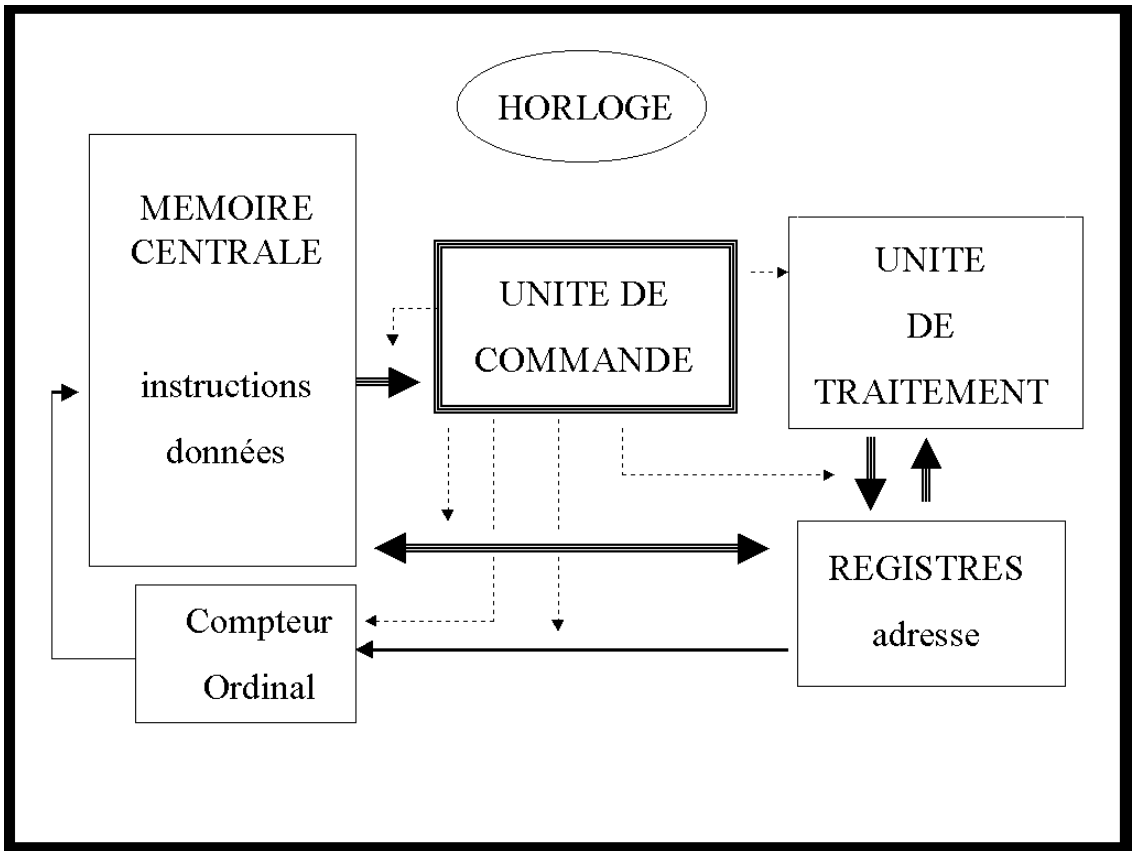
En 1948, F.C. Wiliams et T.Kilburn réalisent à l'Université de Manchester le premier ordinateur à programme enregistré. Appelé **Manchester Mark I**, il fut construit pour tester un tube cathodique à mémoire inventé par Williams. Il s'agissait en fait d'un petit ordinateur.

Un ordinateur à programme enregistré plus puissant, l'Electronic Delay Storage (**EDSAC**) fut développé un an plus tard par Maurice V.Wilkes.

Le premier ordinateur électronique, digne de ce nom, voit le jour en 1950, il s'agit de l'**UNIVAC 1**.

En 1949, Jay Forrester réalise la première mémoire à tores. A partir de 1953, la mémoire constituée d'une grille d'anneaux métalliques magnétiques interconnectés remplacera les tubes à vide peu fiables.

Les ordinateurs actuels ont des possibilités de stockage permanent de l'information : disques, CD-ROM, DVD-ROM, Clé USB... L'ordinateur permet d'aller plus vite, de gérer de très grands volumes d'information et d'automatiser des séquences répétitives de traitement. Il permet un gain de temps et contribue à diminuer les erreurs de traitement.



Les composants du système émettent et tiennent à jour des informations de contrôle dits changement d'états. Dans l'UAL, des bascules d'état signalent le résultat d'une opération : retenue entrante, retenue sortante, dépassement de capacité, résultat nul, signe. Dans la mémoire sont signalés la détection et la correction des erreurs de parité. Le comportement de l'unité de commande tient compte de ces informations.

L'unité de commande, véritable chef d'orchestre, est l'organe chargé de la gestion des flux d'information (contrôle) dans l'ordinateur. Il cadence et génère les opérations de contrôle et les commandes appropriées aux différents composants de façon à assurer la réalisation des opérations élémentaires.

Le temps de base est appelé cycle machine. Il est cadencé par une horloge. Le rôle de l'**horloge** est de synchroniser l'émission des signaux de commande.

Une opération élémentaire est une opération qui est réalisée dans une période de temps donnée inférieure au temps de cycle principal. Exemples : transfert de données entre deux dispositifs câblés, addition dans l'UAL, etc.

Le **séquenceur** est l'organe de l'unité de traitement qui se charge, à partir de l'information d'une instruction, de choisir et de générer la séquence d'opérations de commande élémentaires à générer.

Un **compteur ordinal** contient l'adresse de l'instruction à exécuter. L'instruction est lue, puis décodée (analyse du code opérateur et des adresses des opérandes), les commandes élémentaires sont générées, l'instruction est ensuite exécutée.

L'**UAL** est composée d'un certain nombre d'unités fonctionnelles : addition, multiplication, division, décalage logique de bits vers la gauche ou la droite, etc. Elle possède des éléments de mémorisation appelés registres utilisés en entrée et en sortie des unités fonctionnelles ou pour conserver des résultats temporaires.

Le **BUS de données** permet la transition des données entre la mémoire centrale et l'unité de traitement. Le **BUS d'instructions** permet la transition des instructions entre la mémoire centrale et l'unité de Commande. Une partie de la mémoire centrale peut être dupliquée (cachée) dans une mémoire plus rapide appelée **mémoire cache**, qui peut faire partie ou non du microprocesseur.

La partie matérielle de l'ordinateur (**Hardware**) comprend la partie contrôle électronique et en particulier tout ce qui peut être obtenu par câblage électronique: une opération arithmétique simple comme une addition, une multiplication, etc. La partie logique (**Software**) comprend la partie programmée utilisant les ressources matérielles et électroniques existantes: l'algorithme de calcul d'un cosinus requiert des additions, des multiplications et des divisions.

L'histoire des micro-ordinateurs est différente de celle des calculateurs scientifiques ou de gestion. Les premiers micro-ordinateurs voient le jour dans les années 1970. Leur vocation était surtout ludique. Le langage *BASIC* était utilisé pour les programmer.

Ce n'est qu'en 1981 que le premier PC d'IBM voit le jour (4 ans après l'Apple 2). Doté d'un système d'exploitation comme sur les grosses machines, le Personal Computer doit son succès à un programme de bureautique appelé *tableur*.

QUELQUES DATES HISTORIQUES

1642 Machine arithmétique de B. Pascal

1725 Métier à tisser de Jacquard (automate à programmes extérieurs)

1822 Machine différentielle de C. Babbage

1833 Machine analytique qui ne voit jamais le jour

1848 Algèbre de G. Boole

1890 Recensement des Américains (carte perforée de H. Hollerith)

1924 Création d'IBM et de la compagnie des machines Bull

1936 **Church** et **Türing** formalisent la notion d'algorithme.

1938 **Shannon** soutient une thèse sur l'algèbre Booléenne et l'électronique. Il développera par la suite toute la théorie de l'information

1941 Origine de la Cybernétique avec **Waren Mc Culloch**, **Walter Pitts** et **Norbert Wiener** avec la théorie de la commande des asservissements

1943 1ère calculatrice électronique ENIAC (Electronic Numerical Integrator And Computer) : 18000 tubes et logique câblée.

194x Machines de Türing : calculatrices intégrées

1944 **Goldstine** et **Von Neumann** développent la théorie des jeux

1944 **Howard Hathaway Aiken** met au point le premier calculateur commandé par programme l'**ASCC Mark I** (Automatic Sequence Controlled Calculator). Il s'inspire des plans de la machine analytique de Charles Babbage.

1946 **Von Neumann** introduit le concept de programme enregistré dans une mémoire: les instructions sont exécutées en séquence et certaines instructions permettent le débranchement C'est à cette date que l'on situe la naissance de l'informatique.

1946 Calculateur électronique **ENIAC** (Electronic Numerical Integrator and Calculator) de **J.Presper Eckert, John Mauckly** et **Goldstine**.

1947 Le transistor

1948 **F.C.Williams** et **T.Kiburn** réalisent le premier ordinateur à programme enregistré: le **Manchester Mark I**.

1948 Calculatrice électronique IBM 604 d'IBM

1950 Premiers ordinateurs UNIVAC 1, IBM et Bull, alliance de l'ordinateur et de la mécanique avec mise au point des machines-outils à commande numériques

1956 Langage de programmation scientifique FORTRAN (IBM)

1957 Fondation de Control Data Corporation (CDC) par Seymour Cray

1959 Informatique à base de transistors (2éme génération)

1960 les circuits intégrés (début 1960)

1960 Mini informatique avec DEC (Digital Equipment Corporation) avec le PDP8

196x COBOL : COmmon Business Oriented Language (gestion)

1964 Création du langage BASIC par Kemeny et Kurz

1964 IBM360 architecture des ordinateurs (Amdahl), Famille de machines avec le même jeu d'instructions, CDC6600

1970 Microprocesseur 4004 de INTEL (ancêtre du Pentium)

1972 Début des microprocesseurs

1972 Fondation de Cray Research Inc. par Seymour Cray.

1973 Système d'exploitation UNIX (version langage C)

1974 Système d'exploitation CP/M par Gary Kildall opérant sur microprocesseur 8088

1975 Microprocesseur Z80 (clône du 8080), Création de Microsoft par Bill Gates et du BASIC sur Altaïr

1976 Installation du premier CRAY-1S aux USA.

1977 Apple II (8 bits Motorola 6502) mémoire de 64 Ko

1978 Microprocesseur 8086 (16 bits), Architecture orientée logiciel (VAX): simplifier la compilation des langages de haut niveau

1979 Le premier tableur : Visicalc sur Apple II

1981 PC d'IBM (16 bits Intel 8088) mémoire inférieure à 1Mo avec le système d'exploitation MS-DOS écrit par Microsoft.

1984 MacIntosh d'Apple avec son interface graphique et sa souris

1984 PC AT (Advanced Technology) processeur 80286 et DOS 3.0

1985 Novembre : Windows 1.01 (PC AT avec HD et écran EGA)

1986 Février : Windows 1.02 version française

1987 PS/2 avec la nouvelle architecture MCA et OS/2 (IBM)

1990 Windows 3.0

1992 Windows 3.1 et OS/2 2.0

1993 Le Pentium d'INTEL

1993 DOS 6.0 avec compresseur de données et antivirus intégré

1995 Windows 95

2. Fonctions d'un système d'exploitation

Le Système d'Exploitation (Operating System) est l'ensemble des logiciels effectuant la gestion optimale des ressources d'un système informatique. Installé, on dit encore « chargé » en mémoire centrale lors du démarrage, le Système Opératoire prend en charge la gestion complexe des constituants d'un ordinateur (processeur, mémoire, périphériques…) et en optimise l'utilisation. Il permet tout simplement de rendre opérationnel un ordinateur.

Les fonctions principales sont:

- gestion des applications ou travaux (partage de l'UC),
- gestion des utilisateurs,
- gestion des interruptions,
- gestion de la Mémoire Centrale,
- gestion de la Mémoire Secondaire (gestion des fichiers).

Un système d'exploitation possède :

- un système de dialogue avec l'utilisateur soit à travers une interface graphique soit à l'aide d'une interface de type « console » où les commandes sont entrées en mode ligne ;
- un interpréteur de commande,
- un certain nombre de services: éditeur de texte, tableur, navigateur, compilateur, etc.

2.1. EVOLUTION HISTORIQUE DES SYSTEMES D'EXPLOITATION

Avec les premières configurations de machines, le système d'exploitation était très primitif. Il était **mono-utilisateur** et **monojob** (monoprogrammé). Les utilisateurs se servaient de l'ordinateur les uns après les autres. Ils préparaient leur travail (job) sous forme de paquet de cartes perforées, puis l'introduisaient dans un lecteur. Le programme et les données étaient enregistrés temporairement dans la mémoire centrale de l'ordinateur. Ils recevaient les résultats sous forme de listings à l'aide de l'imprimante. Il n'y avait pas de mémoire secondaire pour enregistrer les programmes de manière permanente. Un paquet de cartes comprenait donc le programme, les données et des commandes de contrôle pour l'enchaînement et l'exécution du job (JCL).

L'apparition de moyen de stockage (bandes magnétiques, puis disques durs) permit d'enregistrer les programmes et les données. Les paquets de cartes se firent plus petits. Il devint possible d'enchaîner des trains de travaux les uns derrière les autres. Le mode de traitement fut appelé **traitement par lot** ou Batch Processing.

Afin d'optimiser l'utilisation de l'unité centrale qui ne faisait rien lorsqu'un job effectuait des entrées/sorties (E/S ou I/O en anglais), le système d'exploitation devint **multijob** (multiprogrammé). Plusieurs travaux chargés en mémoire centrale se partageaient l'utilisation de l'unité centrale (CPU). Le système d'exploitation veillait à répartir au mieux les ressources CPU, mémoire centrale et mémoire secondaire. Une tranche de temps de l'ordre de quelques secondes était allouée à priori à un job. Lorsque une E/S survenait ou bien à la fin du temps alloué, le CPU était donné au job suivant.

Les moyens de dialogue avec l'ordinateur évoluèrent avec les postes éloignés dits terminaux. Ceux-ci étaient composés d'un écran et d'un clavier. La mise à jour et la préparation des travaux se faisaient en interactif. Les processus utilisateurs étaient gérés par le système d'exploitation comme un traitement particulier prioritaire par rapport au traitement Batch. Le **temps partagé** (Time Sharing) était né et le système d'exploitation devint **multitraitement**. Chaque utilisateur devait se croire seul à utiliser la machine. Une

tranche de temps de l'ordre de quelques centièmes de secondes à quelques dixièmes (selon le nombre d'utilisateurs) était allouée tour à tour à chaque utilisateur. Le traitement par lot s'effectuait en arrière plan lorsque le système n'avait aucun utilisateur à traiter.

Ce type d'architecture était essentiellement centralisé. Les travaux (Batch) étaient mis en file d'attente sur le central et entraient en exécution en plus ou moins grand nombre selon la charge interactive du système. Avec le nombre croissant d'utilisateurs, des machines spécialisées en interactif et en batch vinrent collaborer au sein de cette architecture centralisée. Les terminaux reliés à la machine interactive (Front-End) servaient à la préparation et la soumission des travaux exploités sur la machine Batch (Host). Par la suite, l'architecture centralisée fut remplacée par une architecture distribuée. Les moyens de calcul, de stockage et d'impression jusqu'alors centralisés se sont retrouvés chez les utilisateurs. Le terminal laissa la place au poste de travail (Work Station).

Le système d'exploitation permit aux utilisateurs d'avoir plusieurs processus interactifs associés à une session de travail. Il fut possible de lancer en arrière plan (background) dans une session interactive des processus de compilation, d'impression, etc, tout en continuant une autre tâche (édition par exemple). Le système d'exploitation devint **multitâche** et permit un certain degré de **parallélisme**.

Différents systèmes d'exploitation virent le jour. Chaque constructeur d'ordinateur proposait le sien. Dans les années 1960-70 il ne fallait pas moins de 10 ans pour réaliser un système. Le système Unix écrit en langage C vint révolutionner le monde de l'informatique en 1973. Ce système pouvait s'installer sur n'importe quelle machine en moins d'un an. Il permettait la réalisation rapide d'applications distribuées et assurait le portage de celles-ci sur d'autres machines.

Certains systèmes d'exploitation permettent le traitement temps réel. Les événements extérieurs au système doivent être immédiatement pris en compte. La gestion des interruptions et le partage du CPU est alors plus complexe qu'avec le temps partagé.

2.2. CYCLE D'UTILISATION DE L'ORDINATEUR PAR UN DEVELOPPEUR

La conception d'un programme est la première étape. C'est sans aucun doute l'étape la plus courte, mais y passer du temps permet d'en économiser vraiment par la suite. Viennent ensuite la mise au point du programme et les tests de validation. L'écriture et la modification s'effectuent à l'aide d'un éditeur de texte ou d'un environnement de programmation. Un utilitaire, comme sccs, peut être utilisé pour la gestion des différentes versions d'un programme et la conservation de l'historique des modifications.

Le programme appelé **source** est compilé et donne un fichier **objet**. Le lien du programme objet avec des modules objets des bibliothèques (fonctions et procédures du système ou développées) est effectué par un éditeur de liens (**linker**) pour donner un fichier exécutable. Le fichier exécutable est chargé en mémoire centrale par un chargeur (**loader**). Il existe des compilateurs qui génèrent directement des fichiers exécutables. De même l'édition de liens peut être effectuée au chargement voire même dynamiquement à l'exécution.

Les fichiers objets et exécutables peuvent être générés temporairement juste avant utilisation évitant ainsi l'occupation permanente d'une grande place sur disque. Un compromis est à trouver entre le stockage et le coût de la compilation et de l'édition de liens.

Une fois le programme validé, il est mis alors en exploitation avec des modifications propres à une utilisation particulière (ce qui nécessite une compilation et/ou une édition de liens) et des jeux de données particuliers. La durée d'exploitation d'un travail étant en général de quelques dizaines d'heures, il est nécessaire de tronçonner l'exécution d'un programme et d'effectuer plusieurs passages en machine. En effet l'utilisateur n'est pas

tout seul et il faut diviser le temps machine entre les différents utilisateurs: s'il y a par exemple 24 utilisateurs chacun aura pratiquement droit à une heure par jour. Le tronçonnage des jobs permet en outre à chaque utilisateur de vérifier au fur et à mesure le bon déroulement du programme utilisé (code numérique de simulation).

L'exploitation des codes occupe la plus grande charge de l'ordinateur. La maintenance (détection et correction d'erreurs) et la rénovation des programmes (évolution de l'environnement ou des méthodes numériques) constituent un travail de nature complètement différente.

2.3. LES SYSTEMES D'EXPLOITATION UNIX-LIKE

Le système **Unix** est un système multi-utilisateur, temps partagé et multitâche. Ce système n'a pas été conçu pour le traitement par lot. Néanmoins ce dernier est devenu possible à l'aide des systèmes NQS (Network Queuing System) et RQS (Remote Queuing System) développés par Sterling Software et mis dans le domaine public.

Le système LINUX est un système Unix écrit par Linus Torwald.

Le système GNU/Linux est un système Linux avec un ensemble très conséquent de produits, intégrés au système, mais provenant du monde OpenSource.

Il y a plusieurs interpréteurs de commandes sous Unix ou Linux. Ne faisant pas réellement partis du noyau (kernel) on les appelle des Shells. Exemples : le Bourne Shell (/bin/sh), le C-Shell (/bin/csh), le Korn-Shell (/bin/ksh), le Bourne-Again Shell (/bin/bash)…

2.4. LE SYSTEME D'EXPLOITATION MS-DOS

Le système **MS-DOS** a été conçu pour être mono-utilisateur et monotâche. Certaines commandes comme PRINT peuvent néanmoins s'exécuter à l'arrière plan (background). L'intégrateur graphique Windows permit d'ajouter la gestion et l'environnement multitâche manquants. La version Windows 3.11 peut être considéré comme un cas particulier de système d'exploitation dans un système d'exploitation. En 1995, Windows devint un système d'exploitation à part entière connu sous le nom de Windows 95.

Plusieurs systèmes DOS (Disk Operating System) ont été créés sur PC : IBM-DOS, Digital-DOS, etc.

On parle aujourd'hui du DOS comme d'un interpréteur de commandes. Il a été remplacé pat la commande CMD sous XP et VISTA.

2.5. LES SYSTEMES D'EXPLOITATION WINDOWS

W95, W98, W98ME, W98SE, NT4, NT2000, XP, VISTA

3. Mécanismes de base

3.1. DEFINITIONS

Un **processeur** est l'ensemble des moyens matériels et logiciels permettant l'exécution des instructions. Un **processus** est le déroulement dynamique d'un ensemble d'instructions exécutables sur le même processeur. Un **programme** est un ensemble de processus, éventuellement réduit à un seul élément. Une **ressource** est tout moyen logiciel ou matériel nécessaire au lancement d'un processus, autre qu'un processeur. Les ressources peuvent être une zone de mémoire centrale, de la mémoire secondaire ou un périphérique (dérouleur de bandes, imprimante, …).

Un processus disposant de toutes les ressources nécessaires et d'un processeur adéquat est dit **actif**. S'il lui manque un processeur, il est dit **activable**. S'il lui manque une ressource, il est dit **bloqué** ou en attente de ressource.

3.2. COMMUTATION D'ETAT

Plusieurs processus peuvent se trouver en mémoire centrale en même temps. Ils s'exécutent alors tour à tour. Le passage d'un processus à l'autre doit s'effectuer sans perdre le contexte d'UC du processus en cours (valeurs de l'accumulateur, du registre d'instruction, du compteur ordinal, etc). La sauvegarde du contexte d'exécution permettra de reprendre par la suite l'exécution du processus interrompu. Le mécanisme de base utilisé s'appelle la commutation de mot d'état (eXchange Package). Il permet d'exécuter de manière indivisible la sauvegarde du mot d'état dans une zone précise et le changement d'une nouvelle valeur à partir d'une zone précise.

3.3. INTERRUPTION ET DEROUTEMENT

Le rôle du système d'interruption est d'avertir l'unité centrale qu'un condition particulière est survenue. La prise en compte d'événement (coupure de courant, horloge, E/S, accès mémoire illégaux, instruction illégale, résultat d'opération hors intervalle, etc) doit être instantanée. Le mécanisme d'interruption permet d'interrompre l'exécution d'un processus pour exécuter une routine de traitement appropriée appelée routine d'acquittement (interrupt handler). Le compteur ordinal et le contexte du processus interrompu sont sauvegardés. Le contrôle est transféré à une adresse déterminée et associée à un type d'interruption.

Les interruptions peuvent être dues à des événement matériels ou logiciels. L'exécution du déroutement s'effectue en mode superviseur (par opposition au mode utilisateur).

Le système d'interruption est en général hiérarchisé. A chaque signal d'IT est affecté un niveau de priorité. Si un conflit se présente, l'IT de plus forte priorité sera traitée la première. Il y a en général 3 classes d'interruption:

1) les interruptions externes dont la cause est extérieure au déroulement du processus (panne, intervention de l'opérateur, etc),
2) les déroutements provenant d'une situation exceptionnelle ou d'une erreur liées à l'instruction en cours (division par zéro, adressage illégal, etc),
3) les appels au système (SuperVisor Call ou SVC) permettant de réaliser une fonction du système d'exploitation: consultation des tables système, opération d'E/S, etc (exemple: Intr($13, Reg) en Turbo Pascal sous MS-DOS)

Les principales interruptions sont par ordre de priorité:

- défaillance électrique: lors d'un coupure de courant le système doit sauvegarder le maximum d'information pour garantir la reprise ultérieure des travaux (état de la mémoire centrale, du disque, etc).
- horloge: cette interruption permet de mettre à jour l'horloge interne du système, de partager et de décompter les temps d'utilisation.
- fin d'E/S: le système est prévenu lors de l'achèvement d'une opération sur un périphérique (disque, imprimante, etc).
- protection mémoire: un adressage illégal (operand range error) en dehors des adresses inférieure et supérieure de la zone mémoire alloué au processus provoquera une interruption de l'exécution.
- opération illégale: une division par zéro (division by zero), une opération donnant un résultat non représentable (floating point overflow / underflow), l'exécution d'un instruction inconnue ou de type superviseur provoqueront un arrêt du processus en cours.
- arrivée d'événement temps réel: événements captés par des organes particuliers connectés à l'ordinateur.

L'acquittement d'un signal d'interruption impose qu'un certain nombre de conditions soit rempli:

1) l'UC doit être dans un état tel qu'elle accepte de se dérouter pour un niveau d'interruption donnée, l'UC peut être inhibée ou désinhibée pour une interruption donnée,
2) le niveau de priorité correspondant au signal d'IT doit être armé,
3) aucune IT de niveau de priorité supérieure n'est à l'état actif ou en cours de traitement.

L'UC évolue entre 2 états bien précis vis à vis de chaque interruption: 1) elle met en attente toutes les interruptions d'un certain type (UC inhibée), 2) elle accepte de se dérouter (UC désinhibée).

L'acheminement du signal d'IT doit tenir compte de la priorité de chaque niveau et des niveaux qui doivent être ignorés. L'UC prend connaissance de son état par rapport à une interruption en consultant le mot d'état programme. Ce mot d'état permet de résoudre les problèmes d'incompatibilité entre deux types d'acquittements, de protéger le déroulement des traitements associés à des niveaux moins prioritaires. Certains indicateurs du mot d'état programme (masque d'interruptions) peuvent être positionnés pour retarder temporairement la prise en compte d'une interruption qui sera dite masquée. Il est possible par ailleurs de supprimer complètement la prise en compte d'une interruption en désarmant le niveau de celle-ci.

MASQUER == Retarder la prise en compte d'une IT
DESARMER == Supprimer la prise en compte d'une IT

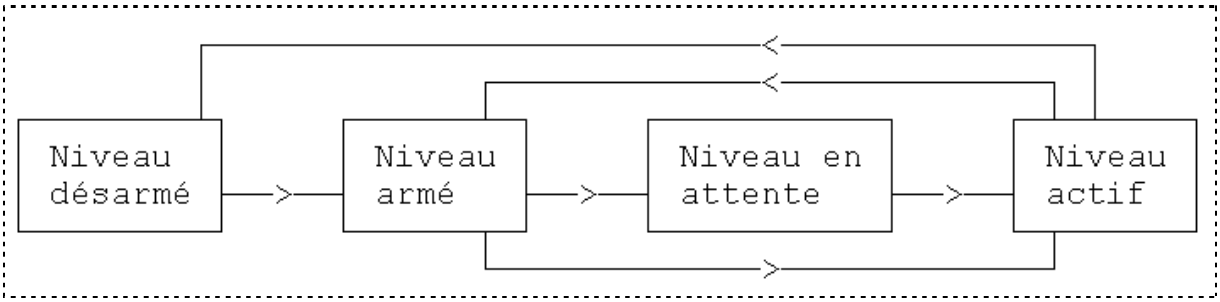
Différents états d'interruption sont possibles:

Niveau armé : le signal sera pris en compte.

Niveau désarmé: le signal d'IT est ignoré et ne sera jamais pris en compte.

En attente : avant déroutement de l'UC, certaines conditions sont à remplir: aucun niveau de priorité supérieur ne doit être actif (en cours), le niveau doit être désinhibé

Niveau actif : la routine d'acquittement du signal est en cours d'exécution.



3.4. DIFFERENCE ENTRE ROUTINE DE TRAITEMENT ET SOUS-PROGRAMME

Le mécanisme d'appel de sous-programme met en jeu un programme appelant et un programme appelé. Avant de donner la main (branchement à l'adresse du sous-programme appelé), l'adresse de retour à l'instruction suivante dans le programme appelant est sauvegardée dans une pile système. S'il y a des variables passées en arguments, les adresses (passage par adresse) ou les valeurs (passage par valeur) sont ajoutées dans la pile. Si des variables locales au sous-programme appelé existent, alors un emplacement mémoire dans la pile leur est éventuellement attribué. Le programme appelé s'exécute, peut communiquer avec le programme appelant à travers des variables globales ou les variables passées en argument, puis rend la main au programme appelé et restaure la dans son état initial (avant appel).

Une routine de traitement n'est pas un sous-programme du programme en cours d'exécution. Elle appartient au système (sauf dans le cas où, par programmation, une autre routine que celle du système peut être proposée par le programmeur). Une interruption arrête l'exécution du programme et donne la main à une routine du système. Le mécanisme de commutation d'état est utilisé.

4. Exemples d'interruption en Turbo-Pascal

4.1. Programme SE_BREAK

Essayer le programme dans l'environnement TURBO-PASCAL et hors environnement (exécution différée à l'aide d'un fichier EXE). Quelles différences peut-on constater? A quel moment est exécutée la routine GestionBREAK? Que se passe-t-il lorsque la boucle devient « Repeat Until false; » ?

4.2. Programme SE_CLICK

Ce programme détourne la routine de traitement de lecture d'une touche enfoncée du clavier. Cette dernière routine est rappelée par la routine de traitement local afin de garantir la lecture de la touche. C'est la nouvelle routine qui émet un léger bip sonore.

Le programme lance un processus CMD.EXE dans lequel on peut relancer la même commande. Appeler plusieurs fois le programme exécutable et constater l'occupation grandissante de la mémoire centrale. Pour cela on utilisera la commande externe « memmap.exe » fournie avec les sources des exercices.

Remarque: L'exécution de l'interpréteur de commande CMD.EXE se termine lorsque la commande EXIT est appelée. Le programme ayant utilisé la commande EXEC se poursuit alors et se termine en remettant en place le traitement antérieur de l'interruption clavier.

4.3. Programme SE_ITIME

Ce programme détourne l'interruption horloge. Appeler plusieurs fois de suite le programme et surveiller l'occupation mémoire. Puis revenir à l'état initial en appelant plusieurs fois la commande EXIT. Décrire ce qui se passe.

4.4. Programme SE_HALT

Ce programme sauvegarde l'adresse « système » de la routine de fin de programme (donnée par la variable ExitPROC) puis définit une nouvelle adresse de fin de programme. La procédure désignée est exécutée en fin de programme quel que soit le déroulement de l'exécution (fin normale, interruption, erreur, etc).

4.5. Programme BGDEMO

A partir des fichiers BGSOUND.PAS, BGDEMO.PAS et du compilateur TPC, créer les fichiers BGSOUND.TPU et BGDEMO.EXE. On utilisera la commande « TPC / ?» pour obtenir la liste les options disponibles de la commande TPC.

Exécuter le programme BGDEMO et l'interrompre par CTRL+BREAK. Que se passe-t-il lorsque la zone précédemment occupée par le programme BGDEMO est remplacée par un autre programme?

5. Exemples de gestion des signaux sous Unix

5.1. LISTE DES SIGNAUX

La liste des signaux peut être obtenue à l'aide de la commande :

> kill -l

```
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGTRAP      6) SIGABRT      7) SIGBUS       8) SIGFPE
 9) SIGKILL     10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     17) SIGCHLD
18) SIGCONT     19) SIGSTOP     20) SIGTSTP     21) SIGTTIN
22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO
30) SIGPWR      31) SIGSYS      32) SIGRTMIN    33) SIGRTMIN+1
34) SIGRTMIN+2  35) SIGRTMIN+3  36) SIGRTMIN+4  37) SIGRTMIN+5
38) SIGRTMIN+6  39) SIGRTMIN+7  40) SIGRTMIN+8  41) SIGRTMIN+9
42) SIGRTMIN+10 43) SIGRTMIN+11 44) SIGRTMIN+12 45) SIGRTMIN+13
46) SIGRTMIN+14 47) SIGRTMIN+15 48) SIGRTMAX-15 49) SIGRTMAX-14
50) SIGRTMAX-13 51) SIGRTMAX-12 52) SIGRTMAX-11 53) SIGRTMAX-10
54) SIGRTMAX-9  55) SIGRTMAX-8  56) SIGRTMAX-7  57) SIGRTMAX-6
58) SIGRTMAX-5  59) SIGRTMAX-4  60) SIGRTMAX-3  61) SIGRTMAX-2
62) SIGRTMAX-1  63) SIGRTMAX
```

Elle peut être aussi obtenue à partir des fichiers *include* :

SIGHUP	Exit	Hangup
SIGINT	Exit	Interrupt
SIGQUIT	Core	Quit
SIGILL	Core	Illegal instruction
SIGTRAP	Core	Trace trap
SIGABRT	Core	Abort
SIGERR	Core	Error exit
SIGFPE	Core	Floating-point exception
SIGKILL	Exit	Kill (cannot be caught)
SIGPRE	Core	Program range error
SIGORE	Core	Operand range error
SIGSYS	Core	Bad argument to system call
SIGPIPE	Exit	Broken pipe
SIGALRM	Exit	Alarm clock
SIGTERM	Exit	Software terminaison signal
SIGIO	Ignore	I/O signal
SIGURG	Ignore	Urgent cond. on I/O chanel
SIGCLD	Ignore	Death of child-process
SIGPWR	Ignore	Power failure
SIGRECOVERY	Ignore	Recovery signal
SIGUME	Core	Uncorrectable memory error
SIGDLK	Core	True deadlock detected
SIGCPULIM	Exit	CPU time limit exceeded
SIGSHUTDN	Ignore	System shutdown imminent
SIGSTOP	Stop	Sendable stop signal
SIGTSTP	Stop	idem from terminal

SIGCONT	Ignore	Continue a stoped process
SIGWINCH	Ignore	Window size change
SIGINFO	Ignore	Information signal (getinfo)
SIGUSR1	Exit	User-defined signal 1
SIGUSR2	Exit	User-defined signal 2

Cette liste est susceptible de changer selon le système d'exploitation.

UTILISATION EN BOURNE SHELL

```
trap 'rm -f /tmp/$$; exit' 1 2 3 15
trap "" 1 2 3 15
```

UTILISATION EN C SHELL

```
onintr label
...
label: command
```

UTILISATION EN LANGAGE C

```
#include <signal.h>
void (*signal (int sig, void (*func) (int))) (int);
void (sigset (int sig, void (*func) (int))) (int);
int sigignore (int sig);
```

Exemple 1

```
#include <signal.h>
main()  {
        void catch(int signal);
        signal(SIGINT, catch);
        /* ... */
}
void catch(int signal)  {
        signal(SIGINT, catch);
        /* ... */
}
```

EXEMPLE 2 AVEC LA FONCTION WAIT:

```
main()  {
int ret_val, ret_stat;
signal(SIGINT, SIG_IGN);
/* forks... */
ret_val = wait(&ret_stat);
/* ... */
}
```