UV2. Système d'exploitation (SE)

Objectif :

L'objectif du cours est de présenter aux étudiants les principales bases d'un système d'exploitation, et de les former à l'utilisation des systèmes les plus répandus. Une partie introductive leur présentera l'historique, les principales composantes d'un système d'exploitation, et les principaux outils de développement que l'on trouve dans un système d'exploitation et la présentation des systèmes les plus répandus : DOS, WINDOWS 95, WINDOWS NT, UNIX.

Contenu :

Module1 : Introduction aux systèmes d'exploitation

1.1 Définition
1.2 Rôle de systèmes d'exploitation
1.3 Historique et types de systèmes d'exploitation
1.4 Nouyau, Processus et Ressources

Module 2 : Système d'exploitation sur PC

2.1 Présentation du BIOS
2.2 DOS
- 2.2.1 Utilisation du DOS
- 2.2.2 Définition d'un repertoire
- 2.2.3 Création d'un repertoire
- 2.2.4 Installation

2.3 Windows 95
- 2.3.1 Présentation
- 2.3.2 Utilisation
- 2.3.3 Installation

2.4 Windows NT
- 2.4.1 Présentation
- 2.4.2 Installation
- 2.4.3 Outils d'administration d'un domaine Windows NT

Module 3 : Système d'exploitation UNIX ou LINUX

3.1 Présentation
3.2 Langage de commande
- 3.2.1 Shell
- 3.2.2 Korn Shell
- 3.2.3 Choix du Shell

3.3 Editeurs de texte
3.4 Ecriture de Scripts et programmation avec un langage de commande (C-shell)
3.5 Compilation séparée
- 3.5.1 Commande Make

MODULE 1 : Introduction aux systèmes d'exploitation

1.1 Définition

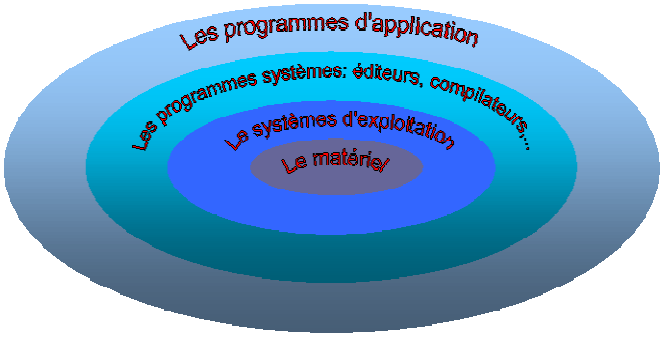
Les systèmes informatiques sont constitués d'un ensemble de composantes matérielles et logicielles.
Le matériel (hardware) comprend :

- Le ou les processeurs
- La mémoire centrale
- Les périphériques d'entrées - sorties :
  - Disque
  - Imprimante
  - Cartes réseaux
  - ….

Les logiciels (Programmes) sont, pour leur part, généralement classés dans deux grandes catégories :

- Les programmes systèmes
- Les programmes d'applications

Les programmes d'applications sont développés pour répondre aux besoins spécifiques des utilisateurs alors que les programmes systèmes contrôlent le fonctionnement des différentes composantes de l'ordinateur.



Le système d'exploitation (operating system en anglais) est la partie la plus importante et la plus connue de ces programmes systèmes voire d'un système informatique tout court. En effet, il est beaucoup plus courant de dire que l'on travaille sur tel ou tel système (UNIX, LINUX, MS-DOS, Windows, OS2, …) plutôt que sur telle ou telle machine.

L'objectif principal des systèmes d'exploitation est tout simplement de rendre l'ordinateur plus facile à exploiter. Il est souvent présentés comme une couche logicielle insérée entre le matériel et les programmes d'applications fournissant ainsi la plate-forme sur laquelle ces programmes sont conçus et implémentés.

L'insertion de cette couche permet de masquer les limitations et les imperfections du matériel en fournissant aux programmeurs le moyen d'accéder aux fonctionnalités offertes par la machine au travers d'une interface beaucoup plus simple à utiliser. Cette couche logicielle permet aussi de proposer de nouvelles fonctionnalités à partir de celles de la machine physique étendant ainsi les possibilités de cette dernière.

1.2 Rôle des systèmes d'exploitation

L'activité principale du système d'exploitation est de gérer les ressources matérielles (processeur, mémoire centrale, imprimante, disque dur, …) en permettant leur allocation et leur partage. Ce dernier point augmente les performances du système en autorisant à plusieurs programmes d'applications d'utiliser simultanément différentes parties de la machine. Il est ainsi pour beaucoup l'interface, construisant pour l'utilisateur, une machine virtuelle plus facile à programmer que la machine réelle.

Le système d'exploitation transforme donc, par sa gestion des ressources matérielles, la machine physique en une machine virtuelle aux capacités infiniment plus grandes que la machine réelle et certainement beaucoup plus souples et plus faciles à exploiter :

- C'est ainsi que la gestion de l'unité centrale assurée par un système temps-partagé va permettre à chacun des utilisateurs de ce système d'avoir l'illusion de posséder chacun un processeur (virtuel) qui travaille exclusivement pour exécuter ses programmes.

- De même, la gestion appropriée de la mémoire centrale permettra :
  - Le chargement de plusieurs programmes à la fois dans cette même mémoire centrale.
  - L'exécution de programmes bien plus grands que la taille de la mémoire centrale

Pour se rendre compte de l'importance du rôle du système en tant qu'interface pour les programmes d'applications, il suffit de considérer le cas d'une machine, fort heureusement irréel, démunie d'un système d'exploitation et sur laquelle nous devons développer quelques programmes d'applications. Imaginez l'effort et le temps, nécessaires à la mise au point des parties des programmes devant interagir avec le matériel et les périphériques. Ces efforts et ces difficultés sont indépendants de la nature de l'application traitée. Et pour se convaincre encore plus, il suffit d'imaginer la quantité de travail de maintenance devant être portée au programme si un de ces périphériques d'entrée-sortie est amené à être remplacé par un autre non nécessairement compatible.

1.3 Historique et types de systèmes d'exploitation

L'histoire et l'évolution des systèmes d'exploitation ont toujours suivi celles de l'architecture des ordinateurs. Ainsi distingue-t-on plusieurs générations dont les plus importantes :

- **Traitement par lot et les systèmes monoprogrammés :** Le système d'exploitation gère un seul programme à la fois. En d'autres termes, quand il commence l'exécution d'un programme, il le termine avant de passer au programme suivant.
- **les systèmes multiprogrammés et temps – partagé** : Le système d'exploitation gère plusieurs programmes. Il commute entre plusieurs programmes.

Il faut y ajouter les systèmes distribués qui ont accompagné l'arrivée des réseaux doit permettre l'éxecution d'un **seul programme** sur **plusieurs machines** distribuer les processus et les remettre ensemble pour gros calculs, p.ex. inversion de grandes matrices d'ordinateurs.

On distingue plusieurs types de systèmes d'exploitation, selon qu'ils sont capables de gérer simultanément des informations d'une longueur de 16 bits, 32 bits, 64 bits ou plus. Alors, nous pouvons essayer de citer: DOS, Windows3.1, Windows95/98/Me, WindowsNT/2000, WindowsXP, Windows7, UNIX / Linux, MAC/OS X, VMS.

-Système d'exploitation nécéssaire

- CP/M (depuis 1974), Digital Research
  - Gestion de disque dur, mais pas d'arborescence
  - Pas de graphisme

  Exemple:
  - CPU 8088, 2 MHz
  - 64 KO de RAM
  - 5 MO de disque dur

- UNIX (depuis 1969-1979), premier par AT&T
  - a servi de modèle pour MS-DOS, Windows.
  - Multi-tâche et Multi-utilisateurs
  - accès simultané aux fichiers, péripheriques, mémoire, processeurs, …
  - Protection mémoire : aucun programme ne peut faire planter le système
  - systèmes de fichiers hiérarchique
  - GUI X-Windows

- MS-DOS (depuis 1981), Microsoft
- MacOS (depuis 1984), Apple
- MacOS (depuis 1984), Apple
  - premier GUI

- Windows (depuis 1991), Microsoft

  Windows 3.11
  - pas de multitâche, pas de multi-utilisateurs

  Windows 95
  - **multi-tâche**
  - premier système 32 bit

  Windows 98
  - Internet integré dans le GUI
  - Plug & Play

  Parallèlement Windows NT
  - système d'exploitation réseaux **multi-utilisateur**

  Windows 2000, et après Windows XP
  - jumellage entre système d'exploitations réseaux et « stand-alone »
- Linux (depuis 1992), OpenSource
  - finlandais Linus Thorwald
  - Licence GPL (General Public Licence) – OpenSource

- Multi-tâche et Multi-utilisateurs
- Distributions
  - Red Hat
  - Fedore
  - S.u.S.e
  - Debian
  - Mandrake...

1.4 Noyau, processus et ressources

1.4.1 Noyau

La majorité des systèmes d'exploitation est construite autour de la notion de noyau ou ***kernel*** (de l'anglais). Le noyau est un programme unique responsable de la communication entre le matériel et le logiciel. En tant que partie du système d'exploitation, le noyau fournit des mécanismes d'abstraction du matériel, notamment de la mémoire, du (ou des) processeur(s), et des échanges d'informations entre logiciels et périphériques matériels. Le noyau autorise aussi diverses abstractions logicielles et facilite la communication entre les processus.
Les noyaux fournissent également des modèles de pilotes et des pilotes pour le matériel.

1.4.2 Processus

Pour comprendre le fonctionnement et la structure des systèmes d'exploitation, il faut commencer par étudier les deux éléments de base : **les processus et les ressources.**

Il s'agit d'une notion introduite avec l'arrivée des systèmes multi-programmés et temps-partagés afin d'éclairer leur fonctionnement.

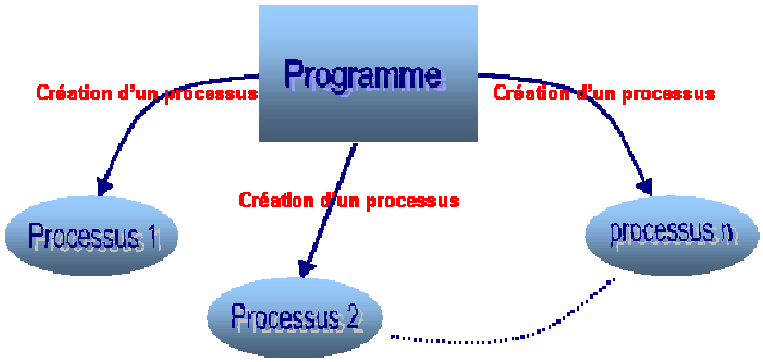
- Un processus se définit comme étant l'abstraction de l'activité du processeur.

  Ou tout simplement

- Comme un programme en cours d'exécution.

Un processus n'est pas un programme exécutable. Il faut savoir distinguer un programme de son exécution. Un processus correspond à une suite d'actions (notion dynamique) réalisées lors d'une exécution. Alors qu'un programme est une suite d'instructions (notion statique).

Le même programme peut s'exécuter plusieurs fois donnant ainsi un processus différent pour chacune de ces exécutions. Ces différents processus peuvent conduire à des résultats différents. Ceci est illustré à la figure suivante.



Exemple :
Pour pouvoir comprendre la différence entre un processus et un programme exécutable, nous allons considérer l'exemple suivant :

Considérons le cas d'un Informaticien préparant un gâteau d'anniversaire pour sa fille :

- Il a une recette
- Il dispose de tous les ingrédients (farine, sucre, œufs, ..)

Dans cette analogie :

- La recette est assimilée à un programme (un algorithme traduit en une suite d'instructions)
- L'Informaticien joue le rôle du processeur (CPU)
- Les ingrédients représentent les données

Comme nous le savons, disposer uniquement de la recette ne suffit pas pour avoir le gâteau. Il faut le préparer selon la recette retenue. *Le processus est l'activité de cet informaticien qui lit la recette, trouve les données et fait cuire le gâteau*.

Le programme (suite d'instructions) est une description précise (mais statique) du comportement à avoir pour résoudre un problème particulier (ce qu'il faut faire et dans quel ordre faut-il les faire pour atteindre l'objectif fixé). Mais, il n'y a rien dans cette description statique qui permet de prévoir (à l'avance) le schéma de déroulement (conditions de déroulement) de l'exécution. Ce schéma peut changer d'une exécution à une autre. Ceci provient des structures conditionnelles, branchements et boucles. En effet une seule branche d'une structure conditionnelle *if-else* est exécutée pendant l'exécution.

Ainsi, dans l'exemple précédent :

- Le fils de l'informaticien peut arriver en pleurant car piqué par une abeille. L'informaticien marque l'endroit où il se trouve dans le processus de préparation du gâteau (l'état du processus en cours est sauvegardé). Cela lui permettra de poursuivre le processus à partir du bon endroit quand il aura fini de soigner son fils : *On dit qu'il y a sauvegarde de contexte du processus de préparation de gâteau* (ou processus cuisine) *ou sauvegarde de l'état du processus de préparation de gâteau.*
- Il cherche un livre sur les premiers soins et commence à soigner son fils.
  Le processeur est passé d'un processus (la cuisine) à un autre (les soins médicaux).

Chacun de ces processus a un programme propre : livre de recettes pour le processus « la cuisine » et livre sur les premiers soins pour le processus « les soins médicaux »

Quand la piqûre est soignée, notre informaticien va reprendre sa recette à l'endroit où il l'avait abandonnée. *On dit qu'il y a restauration de contexte du processus « la cuisine »*.

Le même programme peut s'exécuter plusieurs fois donnant ainsi un processus différent pour chacune de ces exécutions

- L'application de la recette peut être ainsi recommencée plusieurs fois, il s'agit à chaque fois d'une nouvelle activité se terminant par la réalisation d'un nouveau gâteau.
- De même, lors de la réalisation d'un deuxième gâteau tout à fait identique au premier (en déroulant la même recette) certains événements qui ont entravé la première réalisation peuvent ne pas se reproduire (le fils de l'informaticien n'est pas obligé de se faire piquer toujours et encore moins aux même moments de la préparation).

Donc, chaque processus se caractérise par :

- Son programme exécutable
- Ses données
- Sa pile d'exécution.
- Son compteur ordinal. Le compteur ordinal d'un processus est différent de celui de l'ordinateur (l'ordinateur a un seul compteur ordinal qui indique la prochaine instruction qu'il va exécuter). Le compteur ordinal d'un processus indique la prochaine instruction à exécuter dans le programme exécutable de ce processus.

- Son pointeur de sommet de pile de la pile d'exécution. Ce pointeur est utilisé pour faire des empilements (insertion d'un objet au sommet de pile) et dépilements (suppression de l'objet qui se trouve au sommet de pile) dans la pile d'exécution du processus.
- Les autres registres
- Toutes les autres informations nécessaires à l'exécution d'un programme

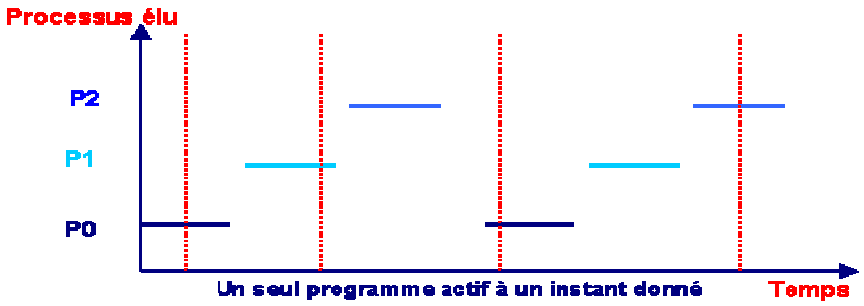
Dans un *système temps partagé*, périodiquement, le système d'exploitation décide d'interrompre un processus en cours et de lancer l'exécution d'un autre, par exemple, parce que le premier a épuisé son temps d'allocation du processeur. C'est ce qui arrive quand on lance l'exécution de plusieurs programmes.

Lorsqu'un processus est temporairement suspendu de cette manière, il faut qu'il puisse retrouver plus tard exactement l'état où il se trouvait au moment de sa suspension. Il faut que toutes les informations dont il a besoin soient explicitement sauvegardées quelque part pendant sa mise en attente :

Si un tel processus possède, par exemple, plusieurs fichiers ouverts, les positions exactes (qui indiquent les endroits où se processus se trouve dans le traitement des fichiers) dans ses fichiers doivent être mémorisées pour qu'une lecture ultérieure qui suit la réactivation du processus lise les bonnes données.

La possibilité qu'a un processeur de commuter d'un programme à un autre en exécutant chaque programme quelques dizaines ou quelques centaines de millesecondes s'appelle *pseudo-parallélisme*.

Dans la figure suivante illustre la commutation du processeur entre trois processus (P0, P1 et P2). Le processeur exécute successivement P0, P1, P2, P0, P1 et P2. Quand un processus est choisi pour être exécuté par le processeur on dit qu'il est élu.



A un instant donné, le processeur n'exécute réellement qu'un seul programme. En revanche, pendant une seconde, il peut exécuter plusieurs programmes. Par exemple, s'il y a dix (10) programmes en cours d'exécution, le processeur peut exécuter chaque programme pendant *1s/10=0.1s*. Comme cette durée (0.1s) n'est pas perceptible par l'homme, cela procure aux utilisateurs l'impression de parallélisme. En vérité, le vrai processeur commute entre plusieurs processus

Etat d'un processus

Dans les systèmes monoprogrammés. Un programme ne quittait pas l'unité centrale avant de terminer son exécution. Pendant cette période, il dispose de toutes les ressources de la machine à lui tout seul. Dans ce type de système, quand le processeur commence l'exécution d'un programme, il termine son exécution avant de passer à l'exécution d'un autre programme. Par contre ce n'est pas le cas dans les systèmes multiprogrammés et temps-partagé.

En effet, comme nous l'avons mentionné dans le paragraphe précédent, il est impossible de prévoir les conditions dans lesquelles va se faire l'exécution d'un programme donné (à cause de l'occurrence de certains événements aléatoires : interruption,….).

Ainsi, durant sa vie, un processus peut être :

- Réellement ***en progression*** (en cours d'exécution),

- ***Suspendu*** en attendant le processeur alloué à un autre processus. Dans un système temps-partagé, périodiquement, le système d'exploitation décide d'interrompre le processus en cours d'exécution et de lancer l'exécution d'un autre, par exemple, parce que le premier a épuisé son temps d'allocation du processeur.

- ***En attente*** d'éléments dont il a besoin et qui ne sont pas disponibles : attente d'une lecture clavier, attente de la fin d'une impression, attente de données provenant d'un autre processus

Par conséquent, la notion d'état de processus est introduite par les systèmes d'exploitation (multiprogrammé, temps-partagé) afin de leur permettre, dans leur gestion de processus, de prendre en considération ces situations.

Un processus peut se trouver dans l'un des états suivants :

- ***Elu*** (en cours d'exécution)
- ***Bloqué*** (attente d'un événement pour pouvoir continuer)
- ***Prêt*** (suspendu provisoirement pour permettre l'exécution d'un autre programme)

Remarque : Dans les systèmes réels, nous avons un nombre plus important d'états. Nous y distinguons les différentes causes d'attente. Un processus peut se bloquer lorsqu'il ne peut pas pour une raison logique poursuivre son exécution.

Exemples:
- un processus peut se bloquer en attendant des données. Par exemple attente d'une saisie au clavier.
- Un processus élu (en cours d'exécution) peut être arrêté, même s'il peut poursuivre son exécution : le système d'exploitation décide d'allouer le processeur à un autre processus

L'état d'un processus est modifié sous l'effet d'événements. Les événements sont :

- ***Internes aux processus*** : une demande de lecture (au clavier, fichier) ou d'écriture (dans un fichier) fait passer le processus de l'état élu à l'état bloqué ou «en attente»)
- ***Externes* aux processus** : Ce type d'évènements proviennent du système d'exploitation. Par exemple, l'attribution de l'unité centrale à un processus le fait passer de l'état élu à l'état prêt à l'état élu.

1.4.3 Ressources

Nous appelons ressource toute entité nécessaire à l'exécution d'un processus. Elle est demandée au système par le processus. Si elle est non disponible, alors le système suspend l'exécution de ce processus jusqu'à la disponibilité de cette même ressource. On dit que le processus est bloqué en attendant cette ressource.

Exemples de ressources :
- *Le processeur (dont la référence est implicite).* Le système doit allouer l'unité centrale au processus pour que l'exécution puisse se faire.
- *La mémoire :* Le processus a toujours besoin de la mémoire pour y stocker ses instructions et données. La mémoire peut être allouée statiquement (c'est-à-dire avant le début de l'exécution de ces processus) ou bien dynamiquement à l'exécution (c'est-à-dire pendant l'exécution de ce processus).
- Les fichiers

Module 2. Systèmes d'exploitation sur PC

2.1 Présentation du BIOS

Le BIOS (Basic Input / Output System) traduction en français (système de gestion élémentaire des entrées/sorties) est un programme essentiel de l'ordinateur, permettant le contrôle des éléments matériels. Une partie du BIOS se situe dans le mémoire ROM, cette partie est non modifiable, une deuxième partie du BIOS se situe dans la mémoire EEPROM, mémoire dont le contenu est modifiable. C'est cette partie que l'on modifie lorsqu'on parle du terme "flashage".

Lorsque le système est mis sous-tension ou réamorcé (Reset), le BIOS fait l'inventaire du matériel présent dans l'ordinateur et effectue un test (appelé **POST**, pour "*Power-On Self Test*") afin de vérifier son bon fonctionnement.

- Effectuer un test du processeur (CPU)
- Vérifier le BIOS
- Vérifier la configuration du CMOS
- Initialiser le timer (l'horloge interne)
- Initialiser le contrôleur DMA
- Vérifier la mémoire vive et la mémoire cache
- Installer toutes les fonctions du BIOS
- Vérifier toutes les configurations (clavier, disquettes, disques durs ...)

Si jamais le POST rencontre une erreur, il va essayer de continuer le démarrage de l'ordinateur. Toutefois si l'erreur est grave, le BIOS va arrêter le système et :

- afficher un message à l'écran si possible (le matériel d'affichage n'étant pas forcément encore initialisée ou bien pouvant être défaillant) ;
- émettre un signal sonore, sous forme d'une séquence de bips (*beeps* en anglais) permettant de diagnostiquer l'origine de la panne ;
- envoyer un code (appelé code *POST*) sur le port série de l'ordinateur, pouvant être récupéré à l'aide d'un matériel spécifique de diagnostic.

La plupart des BIOS ont un « setup » (programme de configuration) qui permet de modifier la configuration basique du système. Ce type d'information est stocké dans la mémoire CMOS, contenant tous les paramètres du BIOS autoalimentée (à l'aide d'une pile) afin que l'information soit conservée même lorsque le système est hors tension.
Il existe de nombreux BIOS dans chaque machine :

- Le BIOS de la carte mère
- Le BIOS qui contrôle le clavier
- Le BIOS de la carte vidéo
- et éventuellement
    - Le BIOS de contrôleurs SCSI qui permettent de booter sur le périphérique SCSI, qui communique alors avec le DOS sans pilote supplémentaire
    - (Le BIOS de cartes réseau qui permettent de booter sur le réseau)

Lorsque le système est mis sous tension, le BIOS affiche un message de copyright à l'écran, puis il effectue les tests de diagnostics et d'initialisation. Lorsque tous les tests ont été effectués, le BIOS affiche un message invitant l'utilisateur à appuyer sur une ou plusieurs touches afin d'entrer dans le setup du BIOS.
Selon la marque du BIOS il peut s'agir de la touche *F2*, de la touche *F10*, de la touche *DEL* (sur les claviers français : "*Suppr*"), ou bien d'une des séquences de touche suivantes :

- Ctrl+Alt+S
- Ctrl+Alt+Esc
- Ctrl+Alt+Ins

Réinitialiser le BIOS

Dans la mesure où le setup du BIOS permet de modifier des paramètres matériels, il peut arriver que le système devienne instable, voire ne redémarre plus. Ainsi, lorsque cela arrive, il devient nécessaire d'annuler les modifications apportées au BIOS et de remettre les paramètres par défaut.

Si l'ordinateur démarre et que l'accès au setup du BIOS est possible, celui-ci offre généralement la possibilité de rétablir les paramètres par défaut. Sur les BIOS de type *PhoenixBIOS*, l'appui sur la touche *F9* permet de rétablir les paramètres par défaut du constructeur. Sur les BIOS de type *AwardBIOS*
l'appui sur la touche *F5* rétablit les paramètres précédents, l'appui sur *F6* rétablit les valeurs par défaut du BIOS Award, enfin la touche *F7* permet de rétablir les paramètres par défaut fournis par le contructeur de la carte mère.

Si l'accès au BIOS est impossible par la procédure standard, la plupart des cartes mères sont dotées d'un cavalier (jumper) leur permettant de rétablir les valeurs par défaut. Il suffit de changer la position du cavalier, et de le laisser maintenu dans cette nouvelle position pendant une dizaine de secondes.

2.2 DOS

Le DOS (Disk Operating System) est le premier système d'exploitation né en 1981, utilisé avec les PC. Il a été développé par Microsoft pour la firme IBM et l'ordinateur de type XT. Comme tous les systèmes d'exploitation, DOS est développé pour servir d'interface entre l'électronique d'une part et l'utilisateur d'autre part. Il n'inclut pas d'interface graphique, aussi il est monotâche et monoutilisateur, c'est un système d'exploitation en mode texte utilisant un **prompt**. Le DOS n'est pas intégré en mémoire ROM, mais sur une disquette ou installé sur le disque dur.

Versions du DOS

- DOS 1.00 :1981, il occupe 11 KiB en mémoire et ne gère que des disquettes à simple face de 160 KiB
- PC-DOS 1.10 : 1982, supporte les disquettes 5″¼ doubles faces (320 KiB). Version PC-DOS pour IBM uniquement.
- MS-DOS 1.25 : 1982, équivalent de PC-DOS 1.10 mais en version OEM. C'est la première version vendue par Microsoft (MS-DOS) à d'autres constructeurs.
- DOS 2.00 :1983, version pour les PC XT, il occupe 40 KiB, et sait gérer les disquettes 5″¼ double face de 360 KiB et les disques durs de 15 MiB maximum. Il utilise un système de fichiers en FAT12
- DOS 2.01 : 1983, version internationale du 2.0 ; Support de paramètres localisés, Support du Kanji (caractères japonais).
- DOS 2.1 : 1983, version IBM uniquement. Équipe en particulier le nouveau PC-Jr.
- DOS 2.11 : 1983, support des jeux de caractères spécifiques aux différents pays. C'est une version OEM très largement utilisée par de nombreux constructeurs de compatibles PC.
- DOS 2.25 : 1983, support étendu pour les langues étrangères.
- DOS 3.00 : 1984, version pour le PC AT, occupe 60 KiB, il gère les disquettes 5″¼ de 1,2 MiB et disques durs de 32 MiB, FAT16, il supporte aussi une horloge CMOS.
- DOS 3.05 : 1984, première version OEM pour la version 3.x
- DOS 3.10 : version supportant le réseau
- DOS 3.20 : 1986, version supportant les disquettes 3″½ de 720 KiB, apparition de la commande Xcopy
- DOS 3.30 : 1987, version supportant les disquettes 3″½ de 1,4 MiB
- DOS 3.31 : 1987, supporte des partitions >32 MiB, nouveaux appels systèmes
- DOS 4.00 : 1988, il occupe 110 KiB de RAM, offre une interface graphique (le *Shell*) et gère les disques et fichiers supérieurs à 32 MiB grâce à la FAT16.
- MS-DOS 5 : 1989, meilleure ergonomie ("doskey" pour rappeler/éditer les commandes en ligne voire gérer des "macros") et gestion mémoire (chargement en mémoire haute pour contourner les limitations à 512/640 KiB), interface graphique "DosShell" (apparue avec la version 4) pour une gestion basique des

fichiers et répertoires, cache disque "SmartDrive" améliorant considérablement les performances disque.

- MS-DOS 6.00 :1993, version intégrant un anti-virus, un outil de vérification de système de fichiers, ScanDisk, un logiciel de compression, DoubleSpace, et des menus de démarrage pour gérer plusieurs "configurations".
- MS-DOS 7 :1995, le DOS de Windows 95.
- MS-DOS 7.1 :1996, le DOS de Windows 95B et C et de Windows 98 et 98SE.
- MS-DOS 8 :2000, dernière version de MS-DOS. Il est intégré à Windows ME.
- PC-DOS 2000 : 2000, intègre de petits ajouts de fonctionnalités.

2.2.1 Utilisation DOS

Pourquoi faut-il encore connaître le MS-DOS?
Car il sera votre seul recours si Windows ne se lance pas...

Ces commandes sont tapées à l'invite, c'est-à-dire dans le cas de MS-DOS (Microsoft DOS, le plus connu) une lettre d'unité suivie d'une barre oblique inverse (antislash), ce qui donne A:\ ou C:\ par exemple.

CLS (Clear Screen, i) : Cette commande sert pour effacer l'écran et positionner le curseur en haut a gauche .

Lorsque je suis sur le disque dur (C)

C: \> CLS (Entrer)

Lorsque je suis sur le disquette (A)

A: \> CLS (Entrer)

Lorsque je suis sur le CD-ROM (D)

D: \> CLS (Entrer)

CHANGEMENT DE L'UNITÉ (i) : Les commandes de changement de l'unité dans MS-DOS sont facilement formules en tapant le nom de l'unité (Lettre correspondante) suivi du caractère **:**
C: \> A :
A: \> D :
D: \> F :
F: \> C :
C: \> _
Si le lecteur concerné n'est pas prêt (absence de disquette ou de CD-ROM, le système affiche un message d'erreur)

AIDE : Pour obtenir de l'aide (assistance) sur n'importe quelle commande MS-DOS, il suffit de taper la commande (sans paramètres) suivi du commutateur /?
C: \> COPY /?

DATE (i): Cette commande Affiche la date courante du système avec possibilité de modification.

TIME (i) : Cette commande Affiche l'heure courante du système avec possibilité de modification.

VER (i): Cette commande Affiche la version du système d'exploitation.
C:\>VER
Windows 95. [Version 4.00.950]
C:\>

VOL (i): Cette commande Affiche la nom du volume (label + no série) .

C:\>VOL
Le volume dans le lecteur C est DISQUE_DUR2
Le numéro de série du volume est 2717-1300
C:\>

LABEL (e): Cette commande permet d'attribuer / modifier ou supprimer enlever la nom d'un volume.
C:\>LABEL A:
Le volume dans le lecteur A n'a pas de nom
Nom de volume (11 caractères, si aucun : appuyez sur ENTREE)? DISQUETTE2

C:\>VOL A:
Le volume dans le lecteur A est DISQUETTE2

CHKDSK (e): Cette commande affiche l'état de l'unité, vérifie les erreurs et affiche l'espace libre.

PROMPT (i): Cette commande permet de modifier le prompt du système. possibilité d'utilisation des symboles réservés suivants : $P (unité), $G (Chemin), $T(temps), $D(date) …
C:\>prompt bonjour dt
bonjour Lun 15/11/199912:12:22,70 _
DIR (Directory , i) : Cette commande donne la liste des fichiers et des répertoires dans le lecteur / chemin courant. Pour exécuter cette commande, nous procédons de la façon suivante
C:/> DIR (Entrer)
C:/> DIR A :
C:/> DIR A:/TEST
C:/> DIR /S

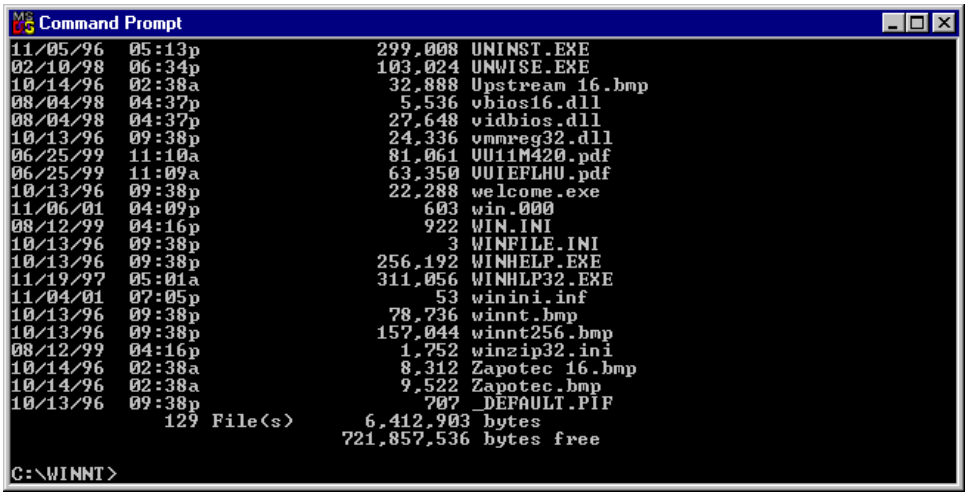


Figure 1 : **La commande** DIR **sans aucune option**

La commande **DIR** avec l'option (/P) donne la liste des fichiers et des répertoires en défilement par page .

Si j'ai plusieurs fichiers dans mon répertoire et si je fais par exemple **C:/> DIR ensuite (Entrer),** je ne peux visualiser que les derniers fichiers que mon écran peut contenir (voir Exemple ci-dessous).

Remarque :

Le réprtoire **WINNT** contient 129 fichiers, mais en faisant **C:/> DIR** ensuite nous tapons **Entrer,** nous avons listé que 21 fichiers que mon écran a pu contenir.

Par contre avec **C:/> DIR /P (Entrer),** je peux visualiser à chaque fois 19 fichiers sur une page (écran) des 129 fichiers. Je peux remarquer sur la figure ci-dessous, que le système me demande à chaque fois de taper une touche pour visualiser le reste des fichiers. En procédant de la même façon jusqu'à la fin.

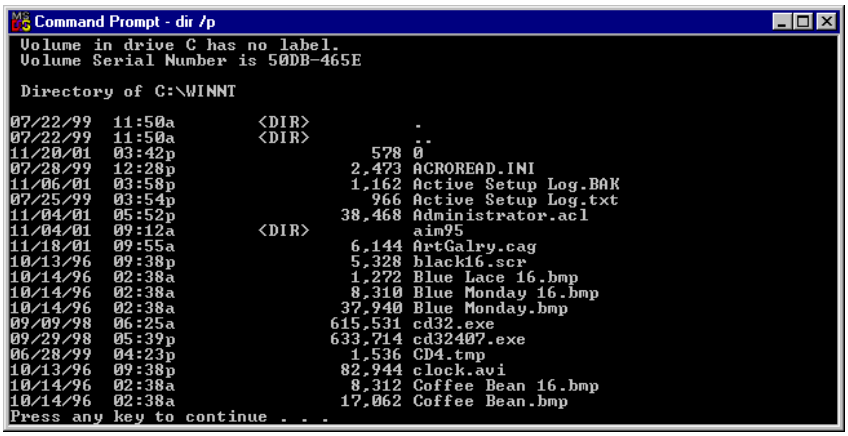


Figure 2 : La commande **DIR** avec l'option **/P**.

La commande **DIR** avec l'option (/W) donne la liste des fichiers et des répertoires partage sur 5 colonnes de l'écran .
Autres commutateurs possibles :
/S : Permet de parcourir toute l'arbre issu du répertoire courant.
/A : Permet d'afficher même les fichiers cachés.
Il est possible d'utiliser une combinaison de ces commutateurs a condition qu'ils soient à la fin de la ligne de commande dir.

Utilisation des caractères génériques * et ? avec dir :

Pour *filtrer* l'affichage sur un groupe restreint de fichiers, le MS-DOS offre la possibilité d'utilisation de deux caractères génériques :
* : signifie n'importe quelle combinaison de caractères.
? : signifie n'importe quelle caractère. Il est à remarquer que plusieurs commandes DOS acceptent ces caractères génériques.

Exemples :
C:/> DIR *.SYS
C:/> DIR ABS*.*
C:/> DIR A?S*.EXE
C:/> DIR ????.COM
C:/> DIR TE??M?.*
TYPE (i) : Cette commande affiche le contenu d'un fichier
C:/> TYPE AUTOEXEC.BAT
C:/> TYPE A:\CONFIG.SYS
C:/> TYPE D:\DEMOS\EXAMPLES.TXT
Pour imprimer le contenu d'un fichier on ajoute le symbole de redirection (>prn) comme suit :
C:/> TYPE AUTOEXEC.BAT >prn

RENAME (i) : Cette commande change le nom d'un fichier. Sa syntaxe générale est : RENAME <ancien_nom> <nouveau_nom>
C:/> RENAME NOMAVANT.AAA NOMAPRES.BBB

DEL (i) : Cette commande supprime (Efface) un ou plusieurs fichiers. Il est a noter que cette commande utilise les caractères génériques pour le cas de plusieurs fichiers. exemples
C:/> DEL BIDON.TXT
C:/> DEL A:*.PAS

C:/> DEL D:\DEMOS\A???.*
C:/> DEL *.* (→ Attention il faut être sur dans ce cas *.*)

COPY (i) : Cette commande copie un ou plusieurs fichiers d'une source a une destination. Il est également nécessaire d'utiliser les caractères génériques pour le cas de plusieurs fichiers. La syntaxe générale de cette commande est :
Copy <Source> <Destination>

Exemples
1- avec destination :
C:/> COPY CONFIG.SYS A:
C:/> COPY AUTOEXEC.BAT AUTO.ORG
C:/> COPY A:*.PAS D:\MYLIB\DEMO
C:/> COPY *.COM A:\REP
C:/> DEL *.* (→ Attention il faut être sur dans ce cas *.*)

2- Sans destination (destination par défaut):
C:/> COPY A:\TEST\MYFILE.TXT

3- Utilisation pour impression :
C:/> COPY A:\TEST\MYFILE.TXT >prn
4- Copie avec fusion (Utilisation de '+') :
C:/> COPY FILE1.TXT+FILE2.TXT+FILE3.TXT FILETOT.TXT
C:/> COPY FILE1.TXT+FILE2.TXT+FILE3.TXT

SYS (e) : Cette commande permet de créer une disquette système (disquette de lancement) . Il permet de transférer les fichiers du noyau ainsi que l'interpréteur de commandes à la disquette.
C:/> SYS A :

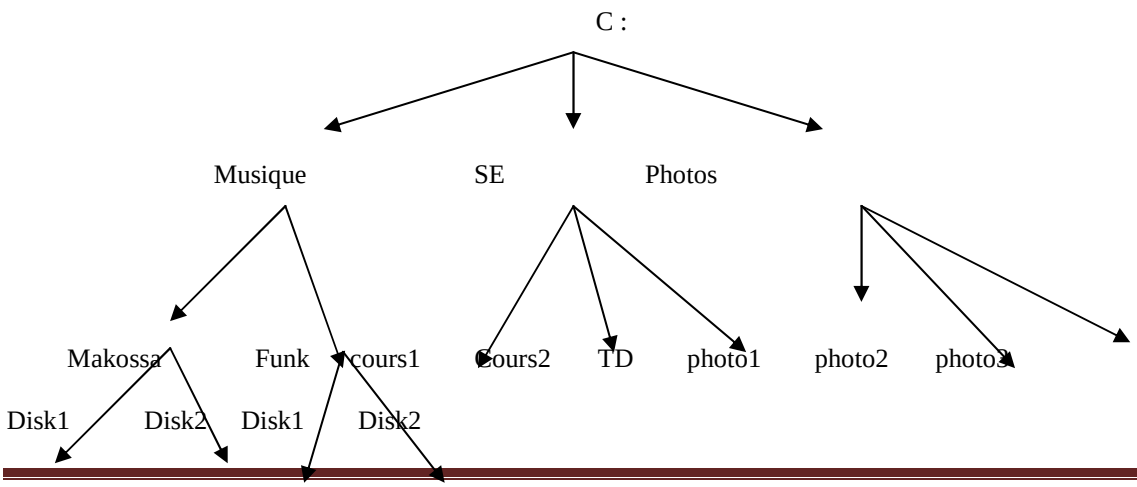
FORMAT (e) : Cette commande permet de formater l'unité concerne. Il est a noter que cette commande supprime tout le contenu antérieur de l'unité avant son formatage. exemples
C:/> FORMAT A :
C:/> FORMAT C : (→ Attention il faut être sur dans ce cas)
Pour formater et créer une disquette système on ajoute le commutateur /S a la commande format .
C:/> FORMAT A : /S

DISKCOPY (e) : Cette commande permet de faire une copie intégrale d'une disquette sur une autre.
C:/> DISKCOPY A: B:

L'utilisateur sera obligé à insérer / retirer chaque disquette dans le lecteur sur plusieurs reprises si la capacité de la mémoire ne permet pas de le faire dans une seule passe.

2.2.2 Définition d'un répertoire

Le disque est comme un classeur qui contient des répertoires et dans le répertoire il y a des sous répertoires



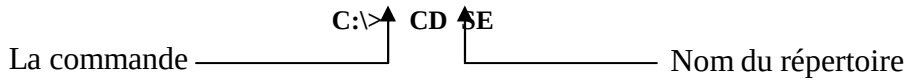
2.2.3 Création d'un répertoire :

La commande **MD** (Make Directory) plus un nom (choisi par l'utilisateur) permet de créer un répertoire. Voir la syntaxe ci-dessous

C:\> MD Math

Entrer dans un répertoire :

La commande **CD** (Current Directory) avec le nom du répertoire (déjà crée) permet d'entrer (changer) dans un répertoire. On procède de la façon suivante



Effacement d'un répertoire :

La commande RD (Remove Directory) avec le nom du répertoire permet d'effacer un répertoire qui existe déjà sur le disque . Deux conditions doivent être satisfaites pour supprimer un répertoire avec la commande RD :

1-On doit être à l'extérieur du répertoire concerné
2-Le Répertoire concerné doit être vide

C:\> RD SE

2.3.4 Installation

Pour installer DOS, il faut s'assurer que les paramètres du BIOS sont configurés de telle façon que le disque s'amorce sur le lecteur A puis sur le disque (boot sequence: A, C). Il faut ensuite mettre la disquette (ou les disquettes selon l'ordre de passage indiqué au cours de l'installation) dans l'unité A, mettre l'ordinateur sous tension, puis suivre les instructions affichées à l'écran.
Il faudra ensuite fournir les informations suivantes :

- l'heure et la date
- le pays de référence
- le support sur lequel le système va être installé (il doit être accessible et comporter suffisamment d'espace libre)
- le répertoire de stockage des fichiers MS-DOS
- le shell MS-DOS doit-il apparaître à chaque démarrage ?
- MS-DOS doit-il être sur une seule partition qui occupe tout le disque dur ?

Création d'une disquette système

Suite à l'installation de MS-DOS il vous faudra créer une **disquette système**.
Après avoir inséré une disquette dans le lecteur, il suffit de taper la commande :

format a: /s

L'argument /s signifie « copier les fichiers systèmes ».

Cette disquette vous permettra de réamorcer le système en l'insérant dans le lecteur au redémarrant l'ordinateur.

2.3 WINDOWS 95

2.3.1 Présentation

Windows 95 remplace la version Windows 3.11 basés sur DOS. La première version est sortie en septembre 1995. Cette version n'est plus supportée par Microsoft depuis décembre 2002.

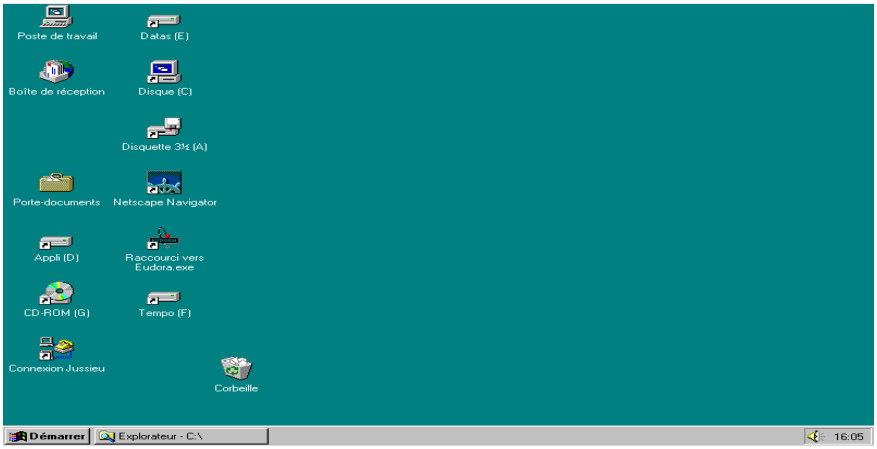
Par rapport à la version Windows 3.11qui finalement n'était qu'une interface graphique basée sur DOS, la version 95 apporte de nombreuses améliorations:

- Système d'exploitation **Multi-tâches**
- utilisation de la **VFAT** qui permet les noms de fichiers longs. La version OSR2 (aussi appelée B) permet l'utilisation de **FAT32**.
- programmation en **32 bits** (du moins en partie), architecture interne remaniée. Il reste compatible avec les programmes 16 bits
- Utilisation de la **base de registre** en remplacement des fichiers config.sys et autoexec.bat (DOS), system.ini et win.ini (Windows 3.X).
- Remplacement des pilotes par des **VXD** (virtual device driver). Ce sont des pilotes 32 bits qui travaillent en mode protégé avec comme extension de noms de fichiers **.VXD**. Les quelques pilotes dynamiques de Win3.X utilisent l'extension .386. Le x représente le type de pilote: VPD (Printer), VDD (Video, carte graphique), ... Ces pilotes sont chargés dynamiquement (uniquement au moment où ils sont nécessaires), à la différence des versions précédentes.
- Modification de l'interface et des menus, avec notamment le bouton Démarrer et le bureau.

Dans la gamme grand-public des systèmes d'exploitation Microsoft, Windows 95 a été suivi par Windows 98 puis Windows Me avant d'être remplacé par la branche Windows NT.

2.3.2 Utilisation

Ouverture
Windows 95 est un système d'exploitation qui sert d'interface graphique entre l'utilisateur et l'ordinateur. Pour lancer Windows 95, mettre l'ordinateur sous tension et attendre d'obtenir l'écran d'accueil qui correspond au "bureau". Ce bureau est personnalisable et peut donc être différent d'un ordinateur à l'autre.

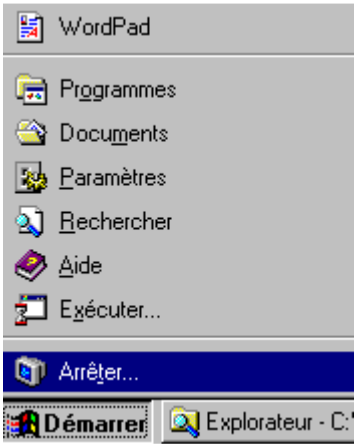


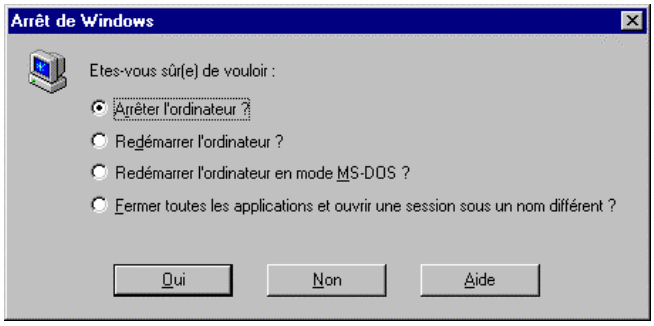
Le bureau comporte différents icônes, par exemple :

- Le poste de travail qui représente le contenu de l'ordinateur (disque dur, disquette, fichier...) ;
- La corbeille pour placer les éléments à supprimer ;
- Des raccourcis qui permettent de lancer des logiciels ou correspondent à des documents ou des dossiers. Les raccourcis se reconnaissent à leur flèche en bas à gauche de l'icône.

La barre des tâches se trouve en général en bas de l'écran. Elle contient, sous forme d'onglet, le menu "Démarrer", les boutons des programmes ouverts en mémoire (ici l'explorateur) et des informations sur le système (ici, l'heure et le réglage du volume sonore). La barre des tâches est parfois cachée et n'apparaît alors que lorsque la souris pointe sur son emplacement.

Fermeture





En fin de travail, cliquer sur le menu "Démarrer", puis cliquer sur le sous-menu "Arrêter..." . La présence des points de suspension indique qu'une boîte de dialogue va s'ouvrir. Cocher la case circulaire devant le choix "Arrêter l'ordinateur" et confirmer la fermeture de l'ordinateur en cliquant sur "Oui". Attendre que l'écran indiquant que l'on peut éteindre l'ordinateur s'affiche avant de fermer les interrupteurs de l'ordinateur.

2.3.3 Installation

L'installation de Windows 95 necessite la disposition d'un CD-rom d'installation qui n'est pas BOOTABLE. Vous devez utiliser une disquette de démarrage pour accéder au lecteur CD-ROM. Cette disquette de démarrage peut être DOS pure, créée par un Windows 95 ou 98. Vous pouvez également créer une partition secondaire et copier l'entièreté du CD d'installation dessus.

Avant d'installer le système d'exploitation, vous devez supprimer la fonction **ChipAway Virus On Guard** dans le BIOS. Sinon, l'installation stoppe avec un écran noir.

a) Création d'une disquette de démarrage

La disquette peut-être créée directement sur un PC avec 95 ou 98 installé.

Dans démarrer > paramètres -> Panneau de configuration, sélectionnons la commande "**Ajout : suppression de programmes et l'onglet** "**disquette de démarrage**". Cette page explique la création sous Windows 98.

Vous pouvez également la créer directement en formatant la disquette avec les fichiers systèmes, en cochant l'option "copier les fichiers systèmes" ou sous DOS par la commande

FORMAT a: /s.

Cette méthode va néanmoins nous obliger à copier manuellement les fichiers nécessaires:

- hymem.sys
- format.com
- un pilote de CD-ROM comme btcdrom.sys par exemple (ils fonctionnent normalement tous)
- le fichier mscdex.exe.

Quelques fichiers vont également être créés pour notre confort:

- format.com
- Fdisk.exe
- keyb.com (pour utiliser un clavier belge ou français).

Il nous reste à créer les fichiers autoexec.bat et config.sys à l'aide d'un éditeur de texte comme edit (toujours fournis dans le dos) ou notepad.

Config.sys
device=himem.sys /testmem:off
device=btcdrom.sys /D:mscd001
éventuellement (pas obligatoire)
files=10
buffers=10
dos=high

Autoexec.bat
LH MSCDEX.EXE /D: mscd001
keyb fr (ou keyb be)

b) Démarrage de l'installation

Le BIOS doit être configuré pour démarrer le PC sur la disquette au préalable. Sur les disquettes crées sous windows automatiquement, un menu permet de sélectionner le démarrage avec ou sans le lecteur CD. Une fois les fichiers de configuration exécutés, DOS va afficher le **prompt**

La première étape va être d'exécuter **FDISK**

A:\ > fdisk

Vous pouvez créer une seule partition ou plusieurs. Si vous créez plus d'une partition, vous devez obligatoirement activer la partition primaire. Une fois la commande terminée, vous devez redémarrer la machine, toujours sur la disquette de démarrage.

A la ligne de commande, tapez D: (ou D est la lettre du lecteur, mais peut être E, F, ...)

En exécutant la commande install.exe, vous démarrez l'installation.

c) Installation de Windows 95 proprement dit

Une fois le programme de démarrage lancé, l'installation passe en mode graphique. La première opération (automatique) est le scandisk (équivalant à la commande CHKDSK) si les partitions sont déjà formatées, formatage préalable sinon. L'installation se fait en 3 étapes:

1. Collecte d'informations sur la configuration de l'ordinateur
2. Copie des fichiers nécessaires
3. Redémarrage de l'ordinateur

La première chose va être de désigner le répertoire d'installation de Win95, c:\Windows par défaut.

La deuxième question va permettre de sélectionner le type d'installation:

1. Par défaut, les composants les plus courants, mais d'autres inutiles voire dangereux comme la compression de disque dur
2. Portable
3. Compacte
4. Personnalisée

Nous allons choisir "personnalisée", ceci va nous permettre plus tard dans l'installation de sélectionner les composants additionnels à installer.

Une fois tapé le numéro de série fournit avec le CD-Rom, vous devez rentrer votre nom d'utilisateur et l'organisation (optionnelle).

La partie suivante va détecter automatiquement les périphériques installés, même si cette détection peut se faire manuellement.

Il nous reste à sélectionner les composants à installer. Dans les accessoires, vous pouvez par exemple ajouter les jeux. Dans disques durs, supprimez "outils de compression de disque" qui entraîne souvent des pertes de données.

Si une **carte réseau** est détectée, l'installation de Windows 95 vous le signale mais permet également d'entrer le nom de l'ordinateur et le groupe de travail. Une fois la liste des périphériques détectés, l'installation va proposer de créer une **disquette de démarrage**, ce qui n'est pas nécessaire puisque nous en avons déjà une.

Windows débute la copie des fichiers sur le disque dur et redémarre l'ordinateur.

Le reste de l'installation se fait automatiquement.

2.4 WINDOWS NT

2.4.1 Présentation

Une fois mis de coté MsDOS (jusqu'à la version 6.22) et Windows (jusqu'à la version 3.10) deux événements majeurs ont été ajoutés aux systèmes d'exploitation microsoft, la gestion intégrée de la notion de réseaux poste à poste, avec windows worksgroup 3.11, et une structure multi-tâche écrite en code 32 bits avec Windows 95.

Cependant tout réseau ne peut être administré en poste à poste, au-delà de 5 à 10 machines, la gestion des mots de passe et de la cohérence de l'ensemble (chaque utilisateur ayant des droits et des permission d'accès à définir sur son poste...,) impose une gestion, et windows 95 n'est pas véritablement un multi-tâche préhemptif, que ce soit à cause de vieilles applications 16 bits, ou bien même pour certaines parties de code de windows 95 lui-même.
Par conséquent deux évolutions ont été faites, dans la lignée de windows 95 au niveau de l'interface, mais radicalement différentes au niveau du code, baptisées de NT pour "New Technologie" pour les démarquer de ce qui existait précédemment.
Selon ce que l'on recherchera, un système d'exploitation 32 bits multi-tâche, ou un système de serveur on utilisera Windows NT Workstation ou NT Server
On peut voir un parallèle entre windows 95 et NT et affiner les différences entre NT Station et NT Server sur les tableaux suivant.

Domaine technique	Windows 95	Windows NT
Configuration *recommandée*	486 avec 8 Mo de RAM et plus (12 recommandé)	Processeur Pentium® (pour les plates-formes Intel) 16 Mo de RAM et plus (24 recommandé)
Compatibilité logicielle	Supporte la plupart des applications Windows 16-bits et MS-DOS®	Ne supporte pas les applications qui violent la sécurité de Windows NT (toute application accédant directement au matériel)
Compatibilité avec le matériel	Supporte plus de 4 000 périphériques	Supporte environ 3 000 périphériques
Installation & déploiement	Code de détection et configuration dynamique des périphériques, permettant une reconnaissance, un configuration et un support du matériel précis (Plug & Play)	Détection et configuration statiques des pilotes de périphériques.
Gestion de l'alimentation et Plug and Play	Support de l'APM pour les portables et de Plug and Play aussi bien pour les mobiles que pour les systèmes de bureau.	Prochaine version.
Performance	Multitâche préemptif pour les applications 32-bits. Performance sensiblement meilleure sur les systèmes à 16Mo.	Multitâche préemptif pour toutes les applications. Performance bien meilleure sur les systèmes à 32Mo.
Fiabilité/Stabilité	Forte amélioration par rapport à Windows 3.11 et Windows pour Workgroups 3.11	Toutes les applications s'exécutent dans un espace mémoire protégé.
Sécurité	Support de la validation des ouvertures de session par le serveur. Sécurité niveau ressource (sécurité niveau utilisateur si connecté à un serveur Windows NT Server)	Protection utilisateur complète jusqu'au niveau des fichiers, même en poste autonome.

2.4.2 Installation

L'installation d'un système d'exploitation se déroule à l'intérieur d'**un programme d'installation,** c'est à dire d'un assistant qui accompagne l'utilisateur pendant toutes les étapes du processus d'installation. Le programme d'installation s'effectue en plusieurs étapes :

- L'analyse de la configuration matérielle et logiciel de l'ordinateur
- La collecte d'information auprès de l'utilisateur
- La décompression et la recopie des fichiers systèmes sur le disque dur de l'ordinateur
- La réinitialisation de l'ordinateur

a) Le choix d'une partition

Le disque dur peut être divisé en plusieurs partitions. Un utilitaire comme **FDISK** permet de diviser un disque dur en plusieurs partitions (l'utilitaire FDISK permet de créer jusqu'à 4 partitions principales ?). Le logiciel PARTITION MAGIC permet de modifier la taille et la structure des partitions d'un disque dur sans avoir à le reformater, et sans perdre les données qui y sont enregistrées.

Une seule partition peut occuper tout l'espace d'un disque dur, mais la division et la séparation de l'espace en plusieurs partitions facilite grandement la gestion des données. Chaque partition peut être spécialisée dans le stockage d'un certain type de fichiers :

- **La partition primaire** qui correspond à l'espace de la première partition d'un disque dur :
  - **La partition active** sur laquelle le programme de démarrage (le chargeur) va chercher la localisation des systèmes d'exploitation (les fichiers d'amorçage).
  - **La partition étendue** qui correspond à l'espace de toutes les autres partitions du disque dur :
    - **Les partitions logiques**
- La partition d'amorçage (qui contient les fichiers d'amorçage du système d'exploitation.
- Les partitions systèmes dédiés aux systèmes d'exploitation (Les fichiers systèmes).
- Les partitions dédiées aux applications (pour diminuer les risques de conflits entre applications, installer plusieurs versions d'un même logiciel,…).
- Les partitions dédiées aux stockages des données (pour faciliter les opérations de sauvegarde,…).

La procédure d'installation de WINDOWS NT permet de partitionner le disque dur.

b) Le choix d'un système de fichier

Chaque partition peut être régie par un **système de fichier** différent (FAT 16, FAT 32, NTFS, EXT2,…) de celui des autres partitions. A chaque système de fichier correspond plus ou moins un système d'exploitation :

- **La FAT 16** sur la partition active permet d'effectuer un démarrage en **multiboot**. La FAT 16 est reconnu à la fois par WINDOWS NT et par WINDOWS 95&98. Le multiboot permet de choisir au démarrage de l'ordinateur lequel des systèmes d'exploitation installés sera chargé en mémoire. La partition active accueille les fichiers d'amorçage des systèmes d'exploitation. Le système DOS limite à 2 Giga Octets, l'étendue d'une partition en FAT 16.
- **La FAT 32** sur une partition logique permet d'optimiser la gestion des fichiers sous WINDOWS 95&98. La partition système accueille les fichiers systèmes de WINDOWS 95&98. Windows NT ne gère pas les partitions en FAT 32.
- **La NTFS** sur une partition logique permet d'optimiser la gestion et le partage des fichiers (les permissions des utilisateurs sont déterminées au niveau des fichiers et non pas au niveau des répertoires) sous WINDOWS NT. La partition système où sont installés les fichiers systèmes de NT.

La procédure d'installation de WINDOWS NT permet de choisir le système de fichier (entre la FAT 16 et NTFS) et de formater le disque dur.

c) La situation de départ

Il peut se présenter trois situation de départ :

- S'il y a un système d'exploitation Windows NT déjà installé, alors il faut utiliser le programme **WINNT32.EXE**.
- S'il y a un système d'exploitation déjà installé, mais qui n'est pas Windows NT, alors, il faut choisir d'installer le système sur une partition formatée en FAT 16 (pour que Windows NT puisse être reconnu par l'autre système d'exploitation. Le système de fichier NTFS n'est reconnu que par Windows NT), et utiliser le programme **WINNT.EXE**.
- S'il n'y a pas de système d'exploitation déjà installé, le disque dur est vierge, et il faut booter avec **les trois disquettes d'amorçage**, puis le CDROM d'installation (ou directement avec le CDROM d'installation, si le lecteur de CDROM est bootable).

d) Le CDROM d'installation de WINDOWS NT SERVER

L'installation du système d'exploitation WINDOWS NT SERVER s'effectue à l'aide du CDROM d'installation. L'installation requière **un accès au lecteur de CDROM**. Le périphérique doit fonctionner et être configurer correctement (avec le pilote adéquat en fonction du système d'exploitation qui va servir le temps de l'installation de WINDOWS NT, en général un système DOS ou le système d'exploitation qui était déjà installé). Le CDROM d'installation de WINDOWS NT a deux fonctions :

- **L'installation**
- **La réparation** d'une installation antérieure

L'installation de WINDOWS NT SERVER se passe dans un premier temps en **mode texte**, puis dans un deuxième temps en **mode graphique**.

e) La procédure d'installation de WINDOWS NT SERVER

La procédure d'installation de WINDOWS NT SERVER se déroule en plusieurs étapes :

La première phase se déroule en mode texte avec le clavier:

- Le programme d'installation propose le choix :
  - **L'installation**
  - **La réparation**
- La recherche des **disques durs** (périphériques de mémoire de masse)
- Les disquettes de certains **pilotes** de périphériques, si nécessaire…
- **L'écran de licence** (Licence Agreement en anglais) qu'il faut lire (PAGE DOWN) et accepter (Touche F8) …
- La partition :
  - **Le partitionnement** du disque dur
  - Le choix du système de fichiers et **le formatage** des partitions
  - **Le répertoire d'accueil** des fichiers système (X:\WINNT par défaut)
  - La vérification de l'état des clusters des disques durs (SCANNING)
- Le premier **reboot**

La deuxième phase se déroule en mode graphique avec la souris:

- L'identification de **l'utilisateur** et de **l'entreprise**
- **Le mode licence** :
  - La licence « par serveur » (Per Server en anglais)
  - La licence « par site » (Per Domain en anglais)
- L'identification de **l'ordinateur** (le nom NetBIOS)
- **Le type d'installation** :
  - Contrôleur Principal de Domaine, le PDC (Primary Domain Controller)
  - Contrôleur Secondaire de Domain, le BDC (Backup Domain Controller)
  - Serveur membre (stand alone server)
- **La saisie du mot de passe** de l'administrateur

- L'enregistrement d'une **disquette de réparation d'urgence** (disquette de boot, ou Emergency Repair Disk en anglais)
- La sélection des **composants du système d'exploitation** (Components en anglais)
- L'installation des **fonctionnalités réseaux**
- La nature du **branchement** de l'ordinateur au réseau
- L'installation de **IIS** (Internet Information Service)
- La configuration de **la carte réseau** :
    - La reconnaissance de la carte :
        - Plug & Play
        - Manuellement
        - L'installation du pilote de la carte réseau
- L'installation des **protocoles réseaux** :
    - Le choix des protocoles (TCP/IP est le protocole par défaut de WINDOWS NT SERVER, c'est un protocole fiable, routable et c'est le protocole d'Internet).
    - La configuration manuelle des paramètres TCP/IP :
        - **L'adresse IP** de l'ordinateur (adresse logique de 32 bits) :
            - L'ID du réseau
            - L'ID de l'hôte dans le réseau
            - **Le masque de sous-réseau** pour identifier la « partie réseau » et la « partie hôte » de l'adresse IP.
            - **La passerelle par défaut du réseau**, pour que les hôtes du réseau puissent communiquer avec les hôtes d'autres réseaux (avec d'autres sous-réseaux IP, ou avec d'autres segments physiques d'un même réseau au sens général). En l'absence de passerelle ou de routeur, les hôtes d'un réseau sont limités aux communications locales (à l'intérieur du sous-réseau IP ou du segment physique).
    - L'installation des **services réseaux**
    - Une boite de dialogue (Intel PROset) affiche **les paramètres de la carte réseau** :
        - L'adresse E/S
        - L'IRQ
        - L'adresse physique
    - Une boite de dialogue (TCP/IP Setup) demande si l'ordinateur utilise un serveur **DHCP** (Dynamic Host Configuration Protocol). L'utilisation d'un serveur DHCP permet la configuration automatique des paramètres TCP/IP (adresse, masque et passerelle). Un serveur DHCP est un serveur WINDOWS NT qui octroie dynamiquement (automatiquement mais pour une durée limitée) les paramètres TCP/IP aux hôtes du réseau (les clients DHCP). Le serveur DHCP configure TCP/IP sur toutes les cartes réseaux qui sont détectées sur l'ordinateur d'un client DHCP…
    - La création et l'affichage des **liaisons** de protocoles
- L'identification du **domaine**
- **L'heure et le jour** (Date and Time Properties)
- **La détection de la carte vidéo**
- **La configuration de l'affichage** (Display Properties)
- **Le deuxième reboot**

Le programme d'installation effectue également les opérations suivantes:

- **L'installation des utilitaires réseaux** (PING, IPCONFIG, NBTSTAT, NETSTAT, ROUTE, TRACERT, ARP,…)
- **L'installation du service WINS** (Windows Internet Name Service). Le service WINS est installé sur un ordinateur dit « client WINS ». Lors du démarrage de la machine, le client WINS se déclare auprès du serveur WINS du réseau, il lui indique son nom d'ordinateur et son adresse IP (les clients WINS s'inscrivent sur le serveur WINS). Quand tous les ordinateurs sont inscrits, chacun peut, par l'intermédiaire du serveur WINS, résoudre les noms NetBIOS des autres ordinateurs (c'est à dire traduire le nom d'une machine en adresses IP). Les utilisateurs connaissent les noms des autres machines, les noms NetBIOS sont conçus pour être compréhensibles par les utilisateurs, ce qui n'est pas forcément le cas d'une adresse IP dynamique.

f) La modification de la configuration après l'installation du système d'exploitation

Pendant, l'installation du système d'exploitation WINDOWS NT SERVER, certains services (le service serveur, le service station,…) sont **installés par défaut** parce qu'ils sont indispensables au bon fonctionnement d'un réseau NT. Le protocole TCP/IP est le protocole réseau par défaut de WINDOWS NT SERVER. Toute fois, il est préjudiciable d'installer tous les services et tous les protocoles s'ils ne sont pas nécessaires, parce qu'ils encombrent la pile de protocole, surchargent la mémoire de travail, et ralentissent le démarrage de l'ordinateur et la connexion au réseau…

L'installation et/ou la suppression d'un protocole, d'une liaison dans la pile de protocole, d'une carte réseau, d'un logiciel client ou d'un service réseaux peuvent s'effectuer après l'installation du système d'exploitation. **Les vérifications et les modifications de la configuration s'effectuent dans le Panneau de Configuration de WINDOWS NT** :

- L'installation d'un protocole supplémentaire peut s'effectuer après l'installation du système d'exploitation
  - **Panneau de Configuration** + **Module Réseau** + **Onglet Protocoles** + **Bouton Ajouter**
- L'activation d'un serveur DHCP peut s'effectuer après l'installation du système d'exploitation
  - **Panneau de Configuration + Module Réseau + Onglet Protocoles + Rubrique Protocole TCP/IP + Bouton Ajouter**
- L'installation d'un service réseau supplémentaire
  - **Panneau de Configuration + Module Réseau + Onglet Services + Bouton Ajouter**
- Le démarrage de certains services peut être automatique ou manuel, un service peut être interrompu, et il est souvent judicieux de vérifier son l'état
  - **Panneau de Configuration + module Services**
- Les liaisons relient les services, les protocoles réseaux (correspondant aux couches hautes du modèle OSI) et les pilotes des cartes réseaux (correspondant aux couches basses du modèle OSI). Les liaisons indiquent le chemin et l'ordre du processus de transmission des données sur le réseau. La première tentative de connexion avec un ordinateur distant s'effectue avec le premier protocole de la liste. L'ordre des protocoles peut être modifié, les protocoles peuvent être montés ou descendus, et il est préférable de disposer le protocole le plus utilisé en début de liste, ou le protocole le plus rapide, ainsi le temps de connexion moyen en sera amélioré. Une liaison peut être rompue, auquel cas il faut vérifier l'état des liaisons
  - **Panneau de Configuration + Module Réseau + Onglet Liaisons**

g) Le mode de licence

L'installation de Windows NT peut s'effectuer sous deux modes de licence différente et exclusive l'un de l'autre :

- **La licence par serveur**. Par exemple, pour un réseau constitué de 2 serveurs NT et de 2 stations NT, il faudra acheter 4 licences.
- **La licence par site** ou par siège. Par exemple, tous les serveurs NT et tous les postes NT qui sont installés sur le même site sont couvert par une seule « licence par site ». Évidemment, la licence par site est beaucoup plus chère qu'une licence par serveur, mais à partir d'un certain nombre de postes, la licence par site est avantageuse.

h) Les composants par défaut de Windows NT

Les composants par défaut de Windows NT sont les suivant :

- **Accessoires**
- **Option d'accessibilité**
- **Multimédia**
- **communication**

i) Les quatre services réseaux de base de Windows NT

Le protocole réseau par défaut est **TCP/IP**.
Les quatre services réseaux de base de Windows NT sont les suivant :

- **Configuration RPC**
- **Interface NetBIOS**
- **Station de travail**
- **Serveur**

j) Le type d'installation de WINDOWS NT SERVER

Avant l'installation du système d'exploitation réseau WINDOWS NT SERVER, l'administrateur doit décider du rôle que jouera le nouveau venu sur le réseau. Le choix du rôle de l'ordinateur est fixé pendant la procédure d'installation. WINDOWS NT SERVER propose et distingue trois rôles :

- **Le Contrôleur Principal de Domaine** (PDC en anglais)
- **Le Contrôleur Secondaire de Domaine** (BDC en anglais)
- **Les serveurs membres**

En fait, l'administrateur doit choisir entre l'installation d'un « contrôleur » ou l'installation d'un « serveur ». C'est un **choix exclusif**, c'est soit l'un, soit l'autre, et c'est un **choix définitif**, il n'est pas possible de transformer un contrôleur en un serveur et vice versa (pour changer le rôle de l'ordinateur, la seule solution est dès lors de procéder à une nouvelle installation).

k) Les domaines de WINDOWS NT SERVER

WINDOWS NT SERVER est organisé autour de la notion de domaine. Tous les ordinateurs d'un réseau NT sont regroupés à l'intérieur d'un ou de plusieurs domaines. Un ordinateur ne peut appartenir qu'à **un et un seul domaine**. Les ordinateurs d'un même domaine peuvent être éloignés « physiquement » les uns des autres, mais appartiennent à une même **« structure logique** ».

Un réseau peut être constitué de **plusieurs domaines**, et c'est souvent bien pratique du point de vue de l'administrateur du réseau. La notion de domaine permet de « segmenter » l'organisation d'un réseau. Chaque domaine est indépendant et souverain, mais des accords peuvent être établis entre les différents domaines NT afin de permettre aux utilisateurs d'un des domaines d'accéder aux ressources d'un autre domaine (c'est le mécanisme de **la relation d'approbation**).

A l'intérieur d'un domaine, il y a obligatoirement un seul Contrôleur Principal de Domaine (CPD). Le Contrôleur Principal de Domaine gère**la liste de tous les utilisateurs du domaine**, avec leurs identifications, leurs droits, leurs permissions et leurs appartenances à des groupes. Les stations gèrent en local **une liste d'utilisateurs locaux** qui peuvent ouvrir une session en local (sans rentrer sur le réseau).

l) Les contrôleurs de domaine WINDOW NT SERVER

Les contrôleurs de domaine WINDOW NT SERVER sont les « chefs » du domaine, mais le Contrôleur Principal de Domaine (le PDC) est le « chef des chefs ».

Lors de la création d'un nouveau domaine, le premier ordinateur de ce nouveau domaine doit être forcément un Contrôleur Principal de Domaine (un **PDC** pour Principal Domain Controller). C'est ainsi que le nom du nouveau domaine et le numéro d'identification du domaine est initialisé. Tous les domaines WINDOWS NT SERVER doivent obligatoirement posséder **un et un seul PDC**. Le rôle d'un PDC est de contrôler et d'authentifier les connexions des utilisateurs au domaine.

Une fois le domaine créé, il est possible d'installer WINDOWS NT SERVER sur d'autres machines en leur assignant le rôle de Contrôleur secondaire de Domaine (**BDC** pour Backup Domain Controller). Un domaine WINDOWS NT SERVER peut posséder **plusieurs BDC**, mais ce n'est pas obligatoire, un domaine peut ne pas avoir de BDC du tout**.**

Les BDC sont une copie du PDC. Les informations contenues dans le PDC (les comptes utilisateurs, la stratégie

de sécurité,…) sont régulièrement transférées aux BDC afin que leur base de données soit continuellement à jour (c'est le mécanisme de **la synchronisation**).

Les BDC, comme le PDC, contrôlent et vérifient les connexions des utilisateurs. Les BDC peuvent être situés sur des segments différents, ce qui permet de répartir les tâches de contrôle sur tout le réseau. Les BDC jouent le rôle de remplaçant du PDC en cas de problèmes. A tout moment un BDC peut devenir le PDC (auquel cas l'ancien PDC devient un BDC, c'est le mécanisme de **la promotion** d'un BDC).

Un contrôleur peut également faire office de serveur, tout dépend de la taille du réseau et de la charge de travail qui est assignée aux contrôleurs. Toutefois, le Contrôleur Principal de Domaine (PDC) est souvent considéré comme un élément stratégique du réseau. L'ordinateur est situé dans un lieu séparé des autres ordinateurs (une chambre blindée par exemple), et les interventions de l'administrateur sur le PDC sont rares et exceptionnelles.

m) Les serveurs membres WINDOWS NT SERVER

L'installation de WINDOWS NT SERVER en tant que serveurs membres permet de fixer le rôle de l'ordinateur au sein du réseau. **Un serveur ne peut pas devenir un contrôleur**. Un serveur peut être dédié à l'exécution de certaines tâches spécialisées :

- Les serveurs de fichiers et d'impression
- Les serveurs d'application
- Les serveurs de messagerie
- Etc…

n) La configuration de la carte réseau

La configuration de la carte réseau peut s'effectuer automatiquement avec la technologie PLUG & PLAY, mais les fournisseurs et les technologies sont nombreux et pas toujours coordonnés. Il est important de s'assurer à l'avance de la compatibilité de la carte réseau :

- **La compatibilité avec le système d'exploitation**
  - Un pilote a été développé et est disponible :
  - Sur le site de l'éditeur
  - Sur le site du constructeur
  - Sur une disquette fournie avec l'équipement
- **La compatibilité avec le support de communication**
  - Le câblage et les connecteurs
  - Le câble coaxial et les connecteurs BNC et/ou AUI (s'il existe une dorsale avec des transceivers externes)
  - Le câble à paire torsadées avec les connecteurs RJ45
- **La compatibilité avec l'architecture interne de l'ordinateur**
  - Les slots de la carte mère :
    - ISA
    - EISA
    - MCA
    - PCI

Le pilote de la carte réseau propose les valeurs par défaut suivantes :

- IRQ : 10
- Port E/S : 300
- Type d'émetteur et de récepteur : le câble coaxial fin

n) L'installation d'un ou de plusieurs protocoles réseaux

L'installation d'un ou de plusieurs protocoles réseaux dépend de la nature du réseau :

- Si le réseau fonctionne avec un seul ou plusieurs protocoles…

- Si le nouvel ordinateur intègre un réseau hétérogène…

Il faut bien sûr installer les protocoles adéquats, c'est à dire installer sur une machine les protocoles qui lui permettra de communiquer avec les autres ordinateurs avec lesquels il est prévu qu'elle communique. Il faut tenir compte de plusieurs facteurs :

- **Les ordinateurs**
  - Par exemple, les ordinateurs APPLE ont besoin d'APPLETALK
- **La segmentation du réseau**
  - Certains segments peuvent être isolés avec un protocole routable
- **Les systèmes d'exploitation réseaux**
  - **Le protocole NetBEUI** ne fonctionne qu'avec les systèmes d'exploitation WINDOWS
  - La coexistence de WINDOWS de MICROSOFT et de NETWARE de NOVELL requière une base commune, qui a été développé par MICROSOFT afin de rendre compatible son système d'exploitation avec les réseaux NOVELL, et ainsi faciliter la migration (ou l'absorption, puisque les parts de marché ont considérablement changé depuis…) :
  - **Le protocole NWLink** qui est la version MICROSOFT de SPX/IPX sur toutes les machines NT qui doivent communiquer avec un serveur NETWARE
  - L'installation du logiciel client NETWARE sur les stations NT :
  - **Le « service client pour NETWARE » (CSNW)** qui permet à un ordinateur WINDOWS NT WORSTATION de se connecter à un serveur NETWARE. CSNW est la version MICROSOFT d'un requêteur NETWARE.
  - L'installation de services sur les stations et les serveurs NT :
    - **Le « service d'annuaire NETWARE » (NDS)** qui permet aux stations NT de profiter des services d'annuaire d'un serveur NETWARE. NDS est la version MICROSOFT du service d'annuaire NDS de NOVELL.
  - L'installation de la passerelle sur les serveurs NT :
    - **Le « service passerelle pour NETWARE » (GSNW)** qui permet à un serveur NT de se connecter et d'émuler un serveur NETWARE. Ainsi, les ressources du serveur NETWARE apparaissent sur le serveur NT. Les clients NT peuvent accéder aux ressources du serveur NETWARE de façon transparente, ils s'adressent au serveur NT sans savoir que celui-ci utilise la passerelle pour atteindre les ressources du serveur NETWARE.
- **Les protocoles existants…**

2.4.3 Outils d'administration d'un domaine Windows NT

Les outils d'administration d'un domaine Windows NT sont les suivant :

- **Le Voisinage Réseau** (une icône placée sur le bureau) pour accéder aux ressources du réseau. Un clic droit, puis l'option PROPRIETES du menu contextuel présente la boîte de dialogue qui correspond au module RESEAU du Panneau de Configuration.
- **Le Poste de Travail** (une icône placée sur le bureau) pour accéder aux ressources locales de l'ordinateur. Un clic droit, puis l'option PROPRIETES du menu contextuel présente la boîte de dialogue qui correspond au module SYSTEME du Panneau de Configuration.
- **L'Explorateur** (avec DEMARRER/PROGRAMMES) pour explorer l'arborescence du disque dur et partager une ressource avec un clic droit, puis l'option PARTAGE.
- **Le Diagnostic NT** (avec DEMARRER/PROGRAMMES/OUTILS d'ADMINISTRATION) pour afficher la version du BIOS (dans Système), le système de fichier (dans Lecteur), le niveau d'accès (dans Réseau).
- **L'administrateur de Disques** (avec DEMARRER/PROGRAMMES/OUTILS d'ADMINISTRATION) pour créer un agrégat de partition. Les agrégats par bande (partitions de même taille) sur plusieurs disques physiques différents permettent d'améliorer les performances en lecture et en écriture. La partition principale (la partition active qui sert à lancer le système d'exploitation au démarrage de l'ordinateur, et qui contient les fichiers systèmes) ne peut faire partie d'un agrégat par bande (ce doit être une seule partition sur un seul disque, par contre, elle peut faire partie d'un système RAID de miroir).
- **Le Gestionnaire des Utilisateurs** (avec DEMARRER/PROGRAMMES/OUTILS d'ADMINISTRATION)

- **L'Observateur d'Evènement** (avec DEMARRER/PROGRAMMES/OUTILS d'ADMINISTRATION) pour visualiser les journaux de sécurité.
- **L'Analyseur de Performance** (avec DEMARRER/PROGRAMMES/OUTILS d'ADMINISTRATION) pour détecter des goulets d'étranglement sur certains matériels.
- **Le Moniteur Réseau** (avec DEMARRER/PROGRAMMES/OUTILS d'ADMINISTRATION) pour capturer des trames, surveiller l'activité du réseau, et programmer des seuils d'alertes.
- **La sauvegarde systématique et régulière des données**. La sauvegarde par segment de réseau indépendant permet de gagner du temps sans perturber l'activité réseau des utilisateurs qui communiquent sur un autre segment.

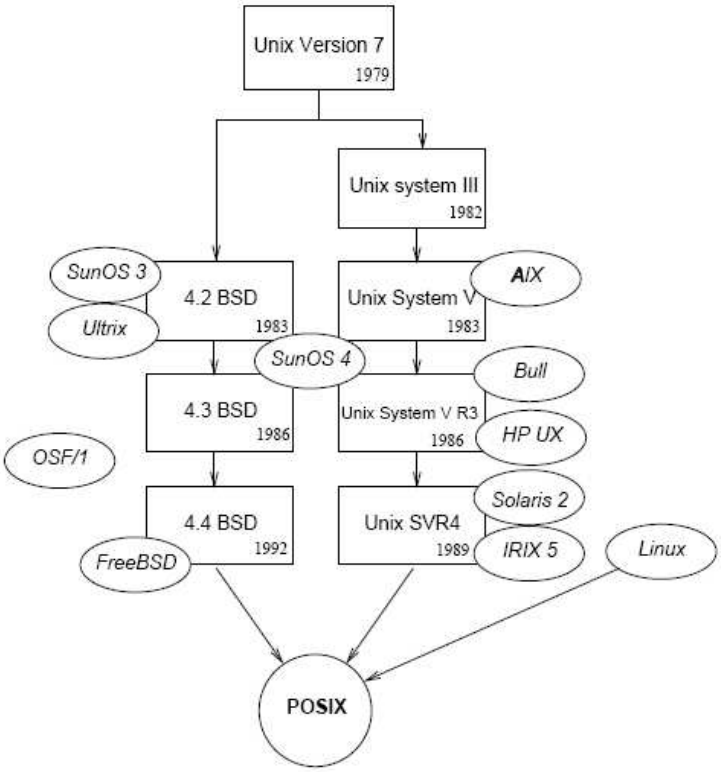
Module 3. Système d'exploitation UNIX ou LINUX

3.1 Présentation

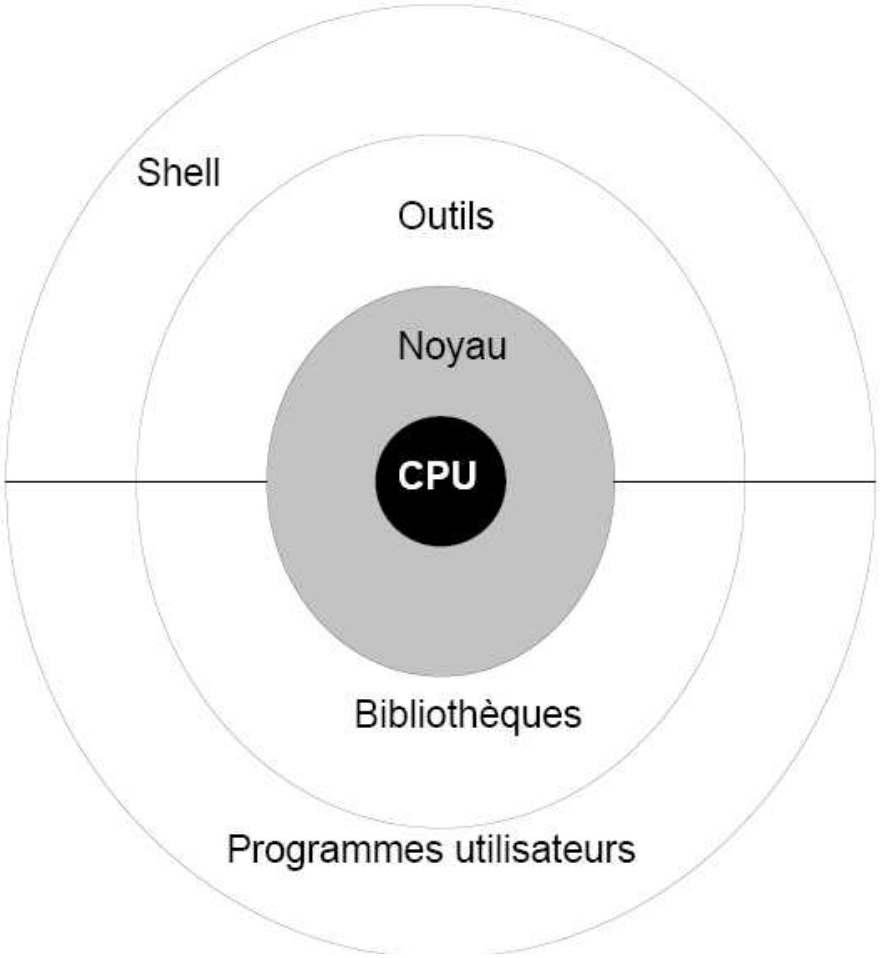
Le système **Unix** est un système d'exploitation multi-utilisateurs, multi-tâches, ce qui signifie qu'il permet à un ordinateur mono ou multi-processeurs de faire exécuter simultanément plusieurs programmes par un ou plusieurs utilisateurs. Il possède un ou plusieurs interpréteurs de commandes (shell) ainsi qu'un grand nombre de commandes et de nombreux utilitaires (assembleur, compilateurs pour de nombreux langages, traitements de texte, messagerie électronique, ...). De plus il possède une grande portabilité, ce qui signifie qu'il est possible de mettre en oeuvre un système Unix sur la quasi-totalité des plates-formes matérielles.

De nos jours les systèmes Unix sont très présents dans les milieux professionnels et universitaires grâce à leur grande stabilité, leur niveau de sécurité élevé et le respect des grands standards, notamment en matière de réseau.

a) Evolution



b) Structure générale



c) Caractéristiques principales

Système d'exploitation
– Multi-tâches en temps partagé
– Multi-utilisateurs
– Interactif
– Intégré aux réseaux
Langages de commande
– Bourne Shell
– Korn Shell
– C-Shell
Plusieurs centaines d'outils.
Outils
-Manipulation de texte
-D´eveloppement de logiciels
-Communication
-Documentation
-Bureautique

d) Le système de fichiers

C'est la partie la plus importante : «Tout est fichier»
On distingue quatre types de fichiers :

- **Ordinaire**, données, programme
- **Répertoire** contient d'autres fichiers ou répertoires
- **Lien symbolique** pointe vers un autre fichier

- **Spécial** permet l'accès à un périphérique

Un fichier est représenté par une structure (I-node) qui stocke les informations sur un fichier : taille, droits d'accès, dates de création, de modification...

Le nom n'est qu'un pointeur sur un I-node.

Un seul type de fichier ordinaire : flot de caractères (8 bits). Les fichiers texte ne sont qu'un cas particulier.

Partage des fichiers en réseau : NFS.

e) Les utilisateurs

Chaque utilisateur du système est identifié par un nom de login auquel est associé :
– un mot de passe
– un identificateur numérique (uid)
– un groupe (gid)
– un commentaire (identité réelle - GCOS)
– un répertoire de travail
– un langage de commandes.
Stockés dans la base de données passwd.
Les groupes permettent aux utilisateurs de partager l'accès à certains fichiers.
Le super-utilisateur root gère tout le système.

f) Les droits d'accès

Trois catégories d'utilisateurs :

le propriétaire	u
le groupe	g
les autres	o

Trois types de droits :

Lecture	r
Écriture	w
exécution	x

Pour un fichier les droits sont exprimés par une chaîne de 10 caractères :

tuuugggooo

t : type du fichier :

Fichier ordinaire	-
Répertoire	d
Lien symbolique	l
fichier spécial	c ou b

uuu : droits du propriétaire
ggg : droits du groupe
ooo : droits des autres

Le super-utilisateur a tous les droits.

g) Accès au système

– Depuis un terminal graphique (Clavier, écran, souris)
– Depuis un terminal ASCII (console) connecté par une ligne série
– Par le réseau

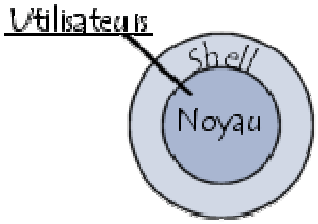
Déroulement :

– Un processus moniteur de port attend des demandes de connexion sur chacune de ces entrées en affichant un message.
– le programme login demande un mot de passe et vérifie sa validité. Il lance le shell dans le répertoire de travail de l'utilisateur.
– Lorsqu'une session de travail est finie, l'utilisateur la termine par la commande logout qui termine le shell et rend la main au moniteur de port.

3.2 Langage de commande

3.2.1 Shell

L'interpréteur de commandes est l'interface entre l'utilisateur et le système d'exploitation, d'où son nom anglais «**shell**», qui signifie «coquille».



Le shell est ainsi chargé de faire l'intermédiaire entre le système d'exploitation et l'utilisateur grâce aux lignes de commandes saisies par ce dernier. Son rôle consiste ainsi à lire la ligne de commande, interpréter sa signification, exécuter la commande, puis retourner le résultat sur les sorties.

Le shell est ainsi un fichier exécutable chargé d'interpréter les commandes, de les transmettre au système et de retourner le résultat. Il existe plusieurs shells, les plus courants étant **sh** (appelé «*Bourne shell*»), **bash** («*Bourne again shell*»), **csh** («*C Shell*»), **Tcsh** («*Tenex C shell*»), **ksh** («*Korn shell*») et **zsh** («*Zero shell*»). Leur nom correspond généralement au nom de l'exécutable.

Chaque utilisateur possède un shell par défaut, qui sera lancé à l'ouverture d'une invite de commande. Le shell par défaut est précisé dans le fichier de configuration */etc/passwd* dans le dernier champ de la ligne correspondant à l'utilisateur. Il est possible de changer de shell dans une session en exécutant tout simplement le fichier exécutable correspondant, par exemple :

/bin/bash

Invite de commande (prompt)

Le shell s'initialise en lisant sa configuration globale (dans un fichier du répertoire */etc/*), puis en lisant la configuration propre à l'utilisateur (dans un fichier caché, dont le nom commence par un point, situé dans le répertoire de base de l'utilisateur, c'est-à-dire */home/nom_de_l_utilisateur/.fichier_de_configuration*), puis il affiche une invite de commande (en anglais **prompt**) comme suit :

machine:/repertoire/courant$

Par défaut dans la plupart des shells le prompt est composé du nom de la machine, suivi de deux points (:), du répertoire courant, puis d'un caractère indiquant le type d'utilisateur connecté :

- «$» indique qu'il s'agit d'un utilisateur normal
- «#» indique qu'il s'agit de l'administrateur, appelé «root»

L'interpréteur de commandes (shell) est un processus.
– Il affiche un prompt, attend la frappe d'une ligne, analyse cette ligne puis exécute la ou les commande(s).
– Pour exécuter une commande il crée un nouveau processus. Il attend la fin du processus créé puis affiche un nouveau prompt.
– Pendant la saisie d'une ligne les touches DELETE ou BACKSPACE permettent d'effacer les caractères saisis.
– Certains shells (Korn Shell, T-CShell) disposent de possibilités interactives plus étendues (historique, complétion. . .).
Le shell dispose de variables et de structures de contrˆole qui en font un langage de commande.

Notion de ligne de commande

Une ligne de commande est une chaîne de caractères constituée d'une commande, correspondant à un fichier exécutable du système ou bien d'une commande du shell ainsi que des arguments (paramètres) optionnels :

ls -al /home/jf/

Dans la commande ci-dessus, ***ls*** est le nom de la commande, ***-al*** et ***/home/jf/*** sont des arguments. Les arguments commençant par **-** sont appelés ***options***. Pour chaque commande il existe généralement un certain nombre d'options pouvant être détaillées en tapant une des commandes suivantes :

commande –help

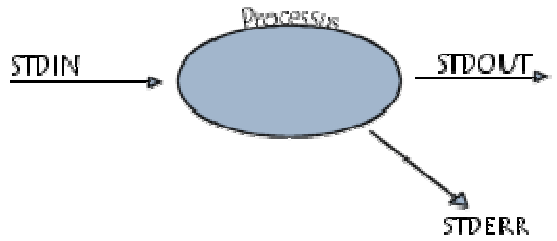
commande -?

man commande

Entrées-sorties standard

Lors de l'exécution d'une commande, un processus est créé. Celui-ci va alors ouvrir trois flux :

- *stdin*, appelé **entrée standard**, dans lequel le processus va lire les données d'entrée. Par défaut *stdin* correspond au clavier ; STDIN est identifié par le numéro 0 ;
- *stdout*, appelé **sortie standard**, dans lequel le processus va écrire les données de sortie. Par défaut *stdout* correspond à l'écran ; STDOUT est identifié par le numéro 1 ;
- *stderr*, appelé **erreur standard**, dans lequel le processus va écrire les messages d'erreur. Par défaut *stderr* correspond à l'écran. STDERR est identifié par le numéro 2 ;



Par défaut lorsque l'on exécute un programme, les données sont donc lues à partir du clavier et le programme envoie sa sortie et ses erreurs sur l'écran, mais il est possible de lire les données à partir de n'importe quel périphérique d'entrée, voire à partir d'un fichier et d'envoyer la sortie sur un périphérique d'affichage, un fichier, etc.

3.2.2 Korn shell

Le korn shell regroupe les fonctions du C shell et du bourne shell, tout en apportant de nouvelles propriétés, afin d'obtenir un shell plus convivial, plus puissant et plus rapide.

Le korn shell a les possibilités supplémentaires suivantes :

- un historique des commandes peut-être mis en place et utilisé (vi ou emacs).
- une compatibilité bourne shell.
- des variables d'environnements supplémentaires (par exemple la définition de directories).
- des commandes Bourne avec de nouvelles fonctionnalités (test, expr, echo).
- des sélections par menu possible.
- des messages d'erreur plus significatifs.
- des alias(Les alias permettent de créer de nouvelles commandes à partir de commandes existantes et d'ainsi établir une bibliothèque personnalisable suivant l'environnement de travail) peuvent être créés.

3.2.3 Choix du shell

Les choix possibles :
- Le Bourne shell /bin/sh (sous AIX : /bin/bsh)
- Le C shell /bin/csh
- Le Korn shell /bin/ksh
- Le Bourne shell again (Bash) /bin/bash

Le positionnement d'un shell peut se faire :
- par l'administrateur (fichier /etc/passwd)
- aprés le login, par une demande utilisateur grace aux commandes :

. *bsh*

. *csh*

. *ksh*

. *bash*

La variable SHELL est affectée lors du login et n'est plus modifiée par la suite (entre autre par les commandes précédentes).
Une fois le sous shell (bsh, csh ou ksh) démarré, il est possible de le quitter à l'aide de la commande ***exit*** ou CTRL D

3.3 Éditeurs de texte

Un **éditeur de texte** est un logiciel destiné à la création et l'édition de fichiers textes (editer : faire apparaitre). Chaque système d'exploitation fournit un éditeur, tant son usage est courant, voire incontournable pour certaines tâches comme l'administration de système et développement logiciel.

Les éditeurs de texte permettent de lire et de modifier les fichiers systèmes et les fichiers de configuration, sous la forme de "**texte brut**". Les éditeurs de texte ne permettent pas de faire de la mise en page, ils ne disposent ni de gras, ni d'italique, ni de souligné, ni de retour à la ligne automatique (il faut appuyer sur la touche ENTREE pour effectuer une "**fin de ligne**").

Les éditeurs de texte présentent leur propre "**prompt** " à partir duquel il est possible de saisir les commandes spécifiques à l'éditeur de texte.
Certains éditeurs de texte proposent plusieurs modes, **le mode de commande et le mode de saisie**.
Des fichiers de configuration peuvent être placés dans le répertoire de base de l'utilisateur afin de personnaliser chaque éditeur de texte.

Il existe de nombreux éditeurs de texte:

- ***ed***
- ***joe***
- ***vi*** *(visual) est un éditeur plein page, qui de ce fait utilise l'ensemble des fonctions disponibles.*
- ***vim*** *(vi improve)*
- ***pico***
- ***emacs*** *(*editor macros)
-

L'editeur « vi » est tres utilisé par les administrateurs, essentiellement pour des raisons pratiques : par exemple, il fonctionne avec n'importe quel terminal.

Les deux principaux éditeurs de texte en mode console sont les suivant:

- **L'éditeur "vi"** est le premier éditeur "plein écran" d'UNIX. "vi" est rapide mais difficile. L'éditeur "vi" fonctionne dans plusieurs "modes". La version améliorée pour LINUX " **vim**" facilite son apprentissage.

    - **Le mode commande** auquel l'utilisateur revient en pressant sur la touche ECHAP.
    - **Le mode insertion** dans lequel l'utilisateur entre en pressant sur la touche "i".

- **L'éditeur "GNU Emacs"** est un **éditeur amodale**, il est plus facile d'emploi, plus facile à assimiler, mais requière plus de manipulation que "vi" pour une même opération. L'éditeur "**xemacs**" fonctionne sous l'interface graphique X Window.

L'éditeur "vi"

L'éditeur "vi" est l'un des premiers éditeurs de texte UNIX, puisqu'il existait à l'époque des **télétypes** qui retranscrivaient sur papier la sortie standard. L'éditeur "vi" est systématiquement installé sur tous les systèmes, c'est parfois le seul éditeur disponible en cas de **dépannage**, et il tient sur une **disquette**. L'éditeur "vi" n'a pas de focntion de mise en forme, ni choix de polices de caractères, ce qui est un avantage pour interpréter les fichiers **scripts** et les fichiers **journaux**.

Si le fichier passé en argument n'éxiste pas, "vi" le crée. L'éditeur "vi"

L'éditeur "vi" démarre toujours en **mode commande** , et il faut passer en **mode édition** pour saisir du texte. Les commandes peuvent être précédées d'un chiffre pour indiquer le nombre de fois qu'elles seront exécutées. Souvent les commandes en majuscule font l'inverse de la même commande en minuscule.

La fenêtre d'aide est parsemée de "**marqueur**" qui sont des expressions encadrées par des barres verticales ("|expression|"). Les marqueurs font références à un emplacement de l'aide consacré à un sujet particulier.

Quelques commandes de l'éditeur "vi"

A partir du mode commande:

ECHAP (pour revenir au mode commande)

i (insert pour passer en mode insertion)

a (passe en mode insertion à droite du curseur)

o (créer une ligne en dessous du curseur)

x (efface le caractère où se trouve le curseur)

dd (efface la ligne du curseur)

u (annule la dernière modification)

CRTL + R (annule la dernière annulation)

G (place le curseur sur la dernière ligne)

1G (place le curseur sur la première ligne)

fx (place le curseur en avant sur la lettre"x")

Fx (place le curseur en arrière sur la lettre "x")

/chaîne (recherche le terme "chaîne")

n (next pour rechercher l'occurence suivante)

p (previous pour rechercher l'occurence précédente)

fx (recherche le "x" suivant sur la même ligne)

Fx (recherche le "x" précédent sur la même ligne)

r (pour remplacer un caractère)

s (pour substituer un caractère à un autre)

y (pour copier la ligne ou se trouve le curseur)

p (paste pour coller le buffer par défaut)

:help (pour obtenir de l'aide)

:q (pour quitter la fenêtre d'aide si elle est ouverte ou l'éditeur)

:q! (pour forcer la sortie de l'éditeur sans enregistrer)

:wq! (pour enregistrer et forcer la sortie de l'éditeur)

:w fichier (pour enregistrer sous le nom fichier dans le répertoire courant)

w (word pour mot suivant)

b (back pour mot précedent)

M (millieu de la page)

{ (paragraphe précédent)

} (paragraphe suivant)

CTRL + F (forward pour avancer d'une page)

CRTL + B (backward pour retourner d'une page)

Fonctionnalités

Les fonctionnalités les plus élémentaires d'un éditeur sont:

- Ouvrir un fichier (en proposant parfois une *liste* de fichiers récemment ouverts, ou déjà existants, voire en permettant de restreindre cette liste par un *filtre*)
- Ajouter du texte dans une ligne, ou des lignes dans un fichier
- Oter des caractères dans une ligne, ou des lignes d'un fichier
- Rechercher/remplacer une chaîne texte (la recherche n'est pas toujours disponible). Un éditeur comme EMACS réalise sa recherche au fur et à mesure de la frappe des caractères, comme les traitements de textes **Wang** à qui cela avait valu sa notoriété. D'autres systèmes attendent la fin de la frappe pour commencer la recherche, à la manière des *mainframes*.
- Sauvegarder le fichier, ou au contraire sortir en renonçant aux modifications (en cas de grosse erreur comme un effacement involontaire de texte).

Exemple : Éditeur pour Windows

- *EDLIN* (MS-DOS, Windows) éditeur ligne par ligne
- *EDIT* (MS-DOS, Windows)
- *Bloc-notes (Notepad)* l'éditeur standard de Windows

3.4 Écriture de scripts et programmation avec un langage de commandes (C-sheel)

Un shell, quel qu'il soit, peut exécuter des commandes prises dans un fichier. Un fichier contenant des commandes pour le shell est appelé un *script*. C'est en fait un *programme* écrit *dans le langage du shell*. Ce langage comprend non seulement les commandes que nous avons déjà vues, mais aussi des structures de contrôle (constructions conditionnelles et boucles).

Pour la programmation du shell, nous allons utiliser le shell sh, qui est le plus répandu et standard. Ce que nous avons vu jusqu'ici s'applique aussi bien à sh qu'à zsh et aussi à csh, à quelques exceptions près, que nous vous signalerons en temps voulu.

a) Écrire un script

Un script shell est un fichier en mode texte. C'est-à-dire que ce n'est pas un fichier binaire, exécutable directement par la machine, mais ***il doit être interprété***.

b) L'interprétation d'un script

L'interprétation signifie que chaque commande contenue dans un script doit être lue par un programme, appelé *interpréteur* ; l'interpréteur analyse chaque commande du script et la traduit en langage machine, ce qui permet l'exécution du script.

On oppose l'interprétation à la compilation, dans laquelle le programme est traduit *une fois pour toutes* en langage machine, quel que soit le nombre de ses exécutions ; tandis que le programme interprété est traduit à la volée pour chacune de ses exécutions. Par exemple, le langage C est un langage compilé.

Dans le cas des scripts shell, l'interpréteur, c'est le shell lui-même. Dans d'autres langages, comme le Perl, l'interpréteur est un programme indépendant du shell.

c) L'édition d'un script

Un script étant un fichier en mode texte, il doit être créé avec un éditeur de texte. Un éditeur de texte est un programme dont la fonction est d'éditer du texte.

Mais quel éditeur choisir ?

Tout d'abord, il faut savoir que **n'importe quel éditeur est capable d'ouvrir et d'écrire des scripts shell**, et vous pouvez tout à fait modifier avec n'importe quel éditeur de texte ce que vous avez écrit avec n'importe quel autre.

Mais il faut savoir aussi que **certains éditeurs de texte sont plus appropriés que d'autres à l'écriture de scripts shell**. Par exemple, ***nano*** permet d'éditer des scripts comme tout autre éditeur, mais quand un script fait plus d'une dizaine de lignes, on commence à s'y perdre un peu. À l'inverse, ***emacs*** et ***vim*** offrent quelques fonctionnalités qui deviennent rapidement indispensables :

- l'indentation ;
- la coloration syntaxique.

L'indentation

L'indentation consiste à « aérer » votre texte selon sa construction logique. C'est très utile, en particulier, quand on a un script qui ressemble à ceci :

```
#!/bin/sh
# Fichier "vote-nir"

echo "Êtes-vous favorable au remplacement du NIR par le VIR ?"
select opinion in Pour Contre
do case $opinion in
# Laisser passer ceux qui répondent correctement à la question
"Pour"|"Contre") break;;
# Au cas où des zozos tapent sur autre chose que 1 ou 2
"*") continue;;
esac
```

```
done

# M'envoyer le résultat par mail
echo "$opinion" | mail bourdieu
```

Même (surtout) si vous ne comprenez pas ce que tout cela veut dire, vous conviendrez que ce n'est pas très lisible. Comparez donc avec ceci :

```
#!/bin/sh
# Fichier "vote-nir"

echo "Êtes-vous favorable au remplacement du NIR par le VIR ?"

select opinion in Pour Contre
  do
    case $opinion in
      # Laisser passer ceux qui répondent correctement à la question
      "Pour"|"Contre") break;;

      # Au cas où des zozos tapent sur autre chose que 1 ou 2
      "*") continue;;
    esac
done

# M'envoyer le résultat par mail
echo "$opinion" | mail bourdieu
```

Les deux scripts sont interprétés exactement de la même façon : l'interpréteur ignore les espaces et les lignes vides. Mais avec l'indentation, on perçoit immédiatement (en tout cas, beaucoup plus vite) la **structure logique** du script.

La coloration syntaxique

Les éditeurs comme ***emacs*** et ***vim*** analysent automatiquement le statut des différents mots et symboles que vous tapez et les colorent logiquement. Par exemple, avec emacs, vous pouvez avoir :

- les commentaires en rouge ;
- les noms de commande en bleu foncé ;
- les noms d'arguments en vert ;
- les noms de variables en jaune ;
- les instructions liées aux boucles en bleu clair ;
- etc.

Ça n'a l'air de rien, dit comme cela, mais comparez vous-même et vous verrez que ces outils sont indispensables, et que l'on y gagne au moins la moitié du temps d'écriture et de débugage.

d) Rendre un script exécutable

Pour que le shell sache comment l'interpréter, un script shell doit commencer par la ligne:

```
#!/bin/sh
```

Il doit aussi avoir être exécutable (bit x). Le #!/bin/sh sur la première ligne indique que ce script doit être exécuté par le shell sh dont on indique le chemin d'accès. Pour rendre un fichier exécutable, tapez :

chaland ~ chmod u+x fichier

Le chemin d'une commande

Quand vous exécutez un script, vous pouvez vous trouver à n'importe quel endroit de l'arborescence de vos répertoires. Si le répertoire courant ne se situe pas dans votre PATH et que vous voulez exécuter un programme qui s'y trouve, vous ne pouvez pas taper :

clipper ~ commande

Car si le répertoire courant n'est pas dans le PATH, le shell n'ira pas y chercher commande.

Vous recevrez donc un message comme :

clipper ~ commande
zsh: command not found: commande

Spécifier le chemin d'une commande

Pour que le shell comprenne où chercher votre commande, il faut donc spécifier l'emplacement de la commande en donnant son chemin, qu'il soit absolu :

clipper ~ /home/toto/repertoire/courant/commande

Ou relatif :

clipper ~ repertoire/courant/commande

Ou encore sous la forme :

clipper ~/repertoire/courant ./commande

Le répertoire ~/bin

Il y a un certain nombre de commandes que l'on peut vouloir utiliser depuis n'importe quel répertoire. Dans ce cas, il est fastidieux de :

- devoir se rappeler dans quel répertoire se trouve chacune d'entre elles ;
- devoir taper à chaque fois le chemin de la commande.

Il suffit donc de mettre tous vos scripts dans un même répertoire, et de mettre ce répertoire dans le PATH. Par convention, ce répertoire s'appelle bin et se place dans votre répertoire personnel. Si votre répertoire personnel est /home/toto, ce répertoire sera donc /home/toto/bin.

Commencez donc par créer ce répertoire :

clipper ~ mkdir bin

Ensuite, vérifiez qu'il soit bien dans votre PATH :

clipper ~ echo $PATH

Si vous voyez par exemple $HOME/bin dans votre PATH, alors c'est bon, tous les fichiers exécutables situés dans ce répertoire seront accessibles depuis n'importe quel répertoire.

Si ce n'est pas le cas, il faut ajouter ce répertoire au PATH. Pour cela, ajoutez dans le fichier de configuration de votre shell, par exemple le fichier .zshrc, la ligne :

PATH=$PATH:$HOME/bin

Cette ligne indique que la prochaine fois que vous ouvrirez votre shell, le répertoire bin figurera dans votre PATH.

e) Principes généraux des scripts shell

- ***Une succession de commandes***

Si vous manipulez déjà le shell en ligne de commande, vous pouvez commencer vos premiers scripts. Un script shell est en effet avant tout une **succession de commandes**.

Par exemple, si vous avez coutume de taper successivement, quand vous vous loguez à l'ENS :

clipper ~ mozilla &

clipper ~ mutt

Vous pouvez vous créer le script suivant dans le fichier ~/bin/amorce :

#!/bin/sh

mozilla &

mutt

Ainsi, dès que vous vous connectez, vous pouvez taper amorce dans le shell, et vos commandes s'exécuteront automatiquement.

- ***Commentaires***

Presque tous les langages informatiques autorisent d'insérer des commentaires ; le shell n'échappe pas à la règle. Pour cela, il suffit de faire précéder chaque ligne de commentaire du caractère « # ». Exemple :

#!/bin/sh

 # Tout ce que j'écris ici ne sera pas lu.
echo "Ce que je tape ici sera lu."

Les lignes de commentaire sont tout bonnement ignorées par l'interpréteur. Alors, allez-vous demander, à quoi servent-elles si elles ne servent à rien ? Elles sont indispensables pour tout programmeur, car elles lui permettent de « commenter » son programme, c'est-à-dire d'écrire ce qu'il veut, comme dans la marge d'un livre.

Les commentaires jouent pour beaucoup dans la lisibilité d'un programme par un humain. Car les lignes de commande pures sont relativement austères ; des commentaires permettent de les décrire à l'intention d'un être humain.

Sans doute allez-vous vous demander quel être humain cela peut bien intéresser. Eh bien, quelqu'un d'autre que vous qui lit ce code ; ou bien vous-même, dans un mois, un an, une décennie, ou plus, quand vous aurez tout oublié de ce programme et de son fonctionnement.

N'hésitez donc pas à recourir abondamment aux commentaires, qui accroissent la lisibilité de votre programme, même s'ils n'ont absolument aucune influence directe sur son fonctionnement intrinsèque.

- ***Lignes blanches***

Les lignes blanches ne sont pas interprétées non plus. N'hésitez donc surtout pas à espacer votre script, les lignes blanches ne consomment presque rien en termes d'espace disque, ce n'est donc pas une ressource rare ; et elles facilitent considérablement la lecture pour un être humain.

- ***L'impératif de lisibilité***

Pourquoi espacer son script ? Pourquoi insérer des commentaires ? Pour une seule et même raison : **votre script doit être lisible**. Pourquoi être lisible ?

D'abord, pour autrui : si d'autres gens lisent votre script, il doit être intelligible, et les passages complexes doivent être explicités par des commentaires.

Ensuite, pour vous-même ; au moment où vous écrivez un script, vous comprenez tout, naturellement ; mais si vous le relisez dans quelques mois, voire quelques années, les passages obscurs risqueront d'être incompréhensibles, ce qui est particulièrement pénible quand on essaye de *débuguer* un programme, c'est-à-dire d'en corriger les erreurs.

Exemples de script

1) Ecrire un programme qui afficher le texte **hello word** : Grâce à la commande **echo**

Faire un **fichier helloworld** contenant les lignes suivantes :

```
#!/bin/sh

# Fichier "helloworld"
echo "Hello world"
```

Exécutez ensuite ce programme, par exemple en tapant, dans le répertoire où il se trouve :

```
clipper ~ $ ./helloworld
Hello world
```

Notre premier programme a été créé.

NB : La commande echo sert à afficher du texte. Chaque ligne de texte est écrite sur une ligne à part.

2) Ecrire u script affichant le texte « **Bonjour Monsieur Comment allez vous ? »**

```
#!/bin/sh
# Fichier "bonjour"

echo "Bonjour  Monsieur "
echo "Comment allez-vous ?"
```

affiche les lignes suivantes :

Bonjour
Comment allez-vous ?

et non :

Bonjour Monsieur Comment allez-vous ?

Annuler le retour chariot

Si vous voulez annuler le retour chariot qui a lieu par défaut à la fin de toute commande echo, il faut utiliser l'option -n. Le programme sera alors :

```
#!/bin/sh

echo -n "Bonjour Monsieur"
echo "Comment allez-vous ?"
```

Alors seulement, vous pourrez avoir :

Bonjour Monsieur Comment allez-vous ?

Citer des variables

Faisons mieux encore : votre script va citer des variables. Pour savoir ce que sont des variables.

La variable USER contient le login de l'utilisateur ; la variable PWD (pour *print working directory*) affiche le répertoire courant. Faisons donc le script suivant :

```
#/bin/sh
# Fichier "mon-pwd"

echo "Bonjour $USER..."
echo "Tu es actuellement dans le répertoire $PWD."
```

Comme vous pouvez le remarquer, pour citer le contenu d'une variable, on ajoute le signe dollar ($) devant son nom.

Exemple. Ecriture d'un script qui écoute : La commande **read**

Prenons le script suivant :

```
#!/bin/sh
# Fichier "mon-nom"

echo "Bonjour… Comment vous-appelez-vous ?"
read nom
echo "Je vous souhaite, $nom, de passer une bonne journée."
```

Vous connaissez déjà la commande echo. La commande read permet de lire des variables. Si vous exécutez ce script, après avoir affiché la ligne

Bonjour …Comment vous-appelez-vous ?

Le shell va attendre que vous tapiez votre nom. Tapez par exemple Albert, puis appuyez sur Entrée, et vous verrez :

Bonjour... Comment vous-appelez-vous ?

Albert

Je vous souhaite, Albert, de passer une bonne journée.
La commande read doit être suivie du seul nom de la variable, **non précédé du signe dollar**. *Le signe dollar ne doit précéder le nom de la variable que lorsque l'on cite son contenu.*

Lire plusieurs variables

La commande read permet également de lire plusieurs variables. Il suffit pour cela d'indiquer à la suite les noms des différentes variables. Exemple :

```
#!/bin/sh
# Fichier "administration"

echo "Écrivez votre nom puis votre prénom :"
read nom prenom
echo "Nom : $nom"
echo "Prénom : $prenom"
```

Vous aurez :

Écrivez votre nom puis votre prénom :
Albert Kengne
Nom: Kengne
Prénom: Victor

Appuyer sur Entrée pour continuer

Nous avons vu comment utiliser read avec un seul argument et avec plusieurs arguments ; il reste à voir l'usage de read *sans* argument. Oui, c'est possible ! Cela équivaut simplement à attendre une réaction de l'utilisateur, mais sans mémoriser ce qu'il tape.

Concrètement, cela est très utile après un message « Appuyez sur Entrée pour continuer. » Exemple :

```
#!/bin/sh
# Fichier "continuer"

echo "Quelle est la différence entre un canard ?"
echo "(Appuyez sur Entrée pour avoir la réponse)"
read
echo "Les pattes, surtout la gauche."
```

3.5 Compilation séparée

La compilation séparée est une caractéristique fondamentale du langage C. Elle permet de diviser un programme en plusieurs modules, chacun étant contenu dans un fichier source dédié. Cela permet de structurer le programme et, dans le meilleur des cas, de pouvoir réutiliser certains modules dans d'autres programmes.
Cependant, dès que le nombre de modules est suffisamment important, il devient fastidieux de taper les commandes de compilation. On aimerait pouvoir disposer d'un outil qui se charge tout seul de la compilation, qui saurait quels modules recompiler et comment le faire. Cet outil magique existe, il s'appelle **make**.

Soit le fichier exécutable nommé **prog**, obtenu à partir de trois fichiers sources, fichier1.c, fichier2.c et fichier3.c et devant être lié avec la bibliothèque mathématique. La compilation manuelle de ce programme s'effectue donc grâce à la commande :

```
$ cc -o prog fichier1.c fichier2.c fichier3.c –lm
```

Outre le fait que cette commande soit longue à taper, cette façon de procéder impose, si l'on ne modifie qu'un seul des trois fichiers C, de compiler également les deux autres, ce qui représente une perte de temps.
Pour améliorer cela, on peut utiliser des fichiers objets. Un fichier objet est le résultat de la compilation d'un fichier C mais sans la phase d'édition de liens. Un fichier objet s'obtient grâce à l'option -c de cc et génère un fichier de même nom que le fichier C mais avec une extension
.o.
En utilisant des fichiers objets, la compilation de prog s'effectue grâce aux commandes :

$ cc -c fichier1.c

$ cc -c fichier2.c

$ cc -c fichier3.c

$ cc -o prog fichier1.o fichier2.o fichier3.o –lm

Ceci est plus compliqué qu'auparavant mais, si l'on ne modifie que fichier1.c, la recompilation de **prog** ne nécessite plus qu'une compilation et une édition de liens, ce qui permet d'éviter deux compilations :

$ cc -c fichier1.c

$ cc -o prog fichier1.o fichier2.o fichier3.o –lm

Cette méthode est donc plus optimale mais nécessite de taper plus de commandes.

3.5.1 Commande make

La commande make heureusement, permet d'automatiser tout cela. Il suffit de lui expliquer comment compiler **prog**, de taper **make**, et la compilation s'effectuera automatiquement en exécutant uniquement les commandes nécessaires.

- **Le fichier Makefile**

Les informations permettant à make de compiler prog doivent se trouver dans un fichier appelé Makefile (ou bien makefile). Un fichier Makefile est un fichier texte. Voici un exemple de fichier Makefile permettant de compiler **prog** :

prog : fichier1.o fichier2.o fichier3.o

⟶ !cc -o prog fichier1.o fichier2.o fichier3.o –lm

fichier1.o : fichier1.

⟶ !cc -c fichier1.c

fichier2.o : fichier2.c

⟶ !cc -c fichier2.c

fichier3.o : fichier3.c

⟶ !cc -c fichier3.c

Un fichier Makefile est composé de règles (il y en a quatre dans notre exemple), qu'on sépare habituellement par des lignes blanches (ce n'est pas obligatoire mais cela permet d'aérer le fichier Makefile et de mieux visualiser les règles).
Chaque règle est de la forme :

cible : dépendance1 dépendance2 ...

⟶ !commande

⟶ !commande

⟶ !...

La première ligne d'une règle indique sa cible, qui est généralement le nom d'un fichier à construire. La cible est suivie d'un deux-points (les espaces autour du deux-points ne sont pas obligatoires mais permettent là encore d'aérer les choses) puis d'une liste de fichiers, appelés dépendances, à partir desquels la cible est construite.

Les lignes suivantes indiquent les commandes (il n'y en a souvent qu'une) à exécuter pour construire la cible à partir de ses dépendances. Chacune de ces lignes doit absolument débuter par une tabulation, représentée dans ce document par une longue flèche. Attention à respecter scrupuleusement ce format car la commande make est très stricte à ce sujet.
Ainsi, la règle :

fichier1.o : fichier1.c

⟶ !cc -c fichier1.c

Signifie que le fichier **fichier1.o** dépend du fichier **fichier1.c**, c'est-à-dire qu'il est obtenu à partir de celui-ci, et que la commande à exécuter pour obtenir
fichier1.o à partir de **fichier1.c** est **cc -c fichier1.c.**

Utiliser la commande make

Une fois le fichier Makefile créé, il suffit de taper make :
$ make

cc -c fichier1.c

cc -c fichier2.c

cc -c fichier3.c

cc -o prog fichier1.o fichier2.o fichier3.o –lm

Les commandes exécutées par make sont affichées au fur et à mesure.

3.5.3 Comment fonctionne la commande make ?

Lorsqu'on l'exécute, la commande make va lire les informations contenues dans le fichier Makefile. Elle considère la première cible rencontrée comme la cible à construire. Dans notre exemple, il s'agit de prog, qui dépend de fichier1.o, fichier2.o et fichier3.o, qui dépendent eux-mêmes respectivement de fichier1.c, fichier2.c et fichier3.c (ces derniers ne dépendent de rien).

Les dépendances de certaines règles peuvent donc être aussi les cibles d'autres règles. On représente habituellement les relations entre cibles et dépendances sous la forme d'un arbre. Notre exemple peut être représenté par l'arbre des dépendances indiqué dans la figure 1.

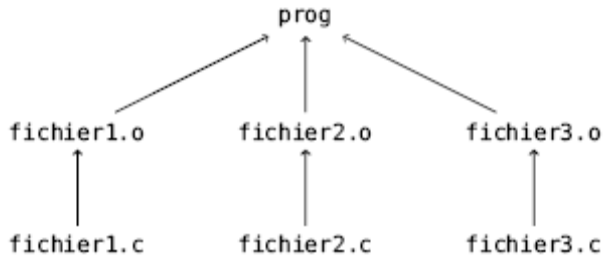


FIGURE 1 – *Arbre des dépendances*

Maintenant, si l'on modifie fichier1.c et lui seul, il faut le recompiler pour obtenir un nouveau fichier1.o. En revanche, fichier2.c et fichier3.c n'ayant pas été modifiés, il n'est pas nécessaire de les recompiler. Enfin, il faut refaire l'édition de liens de fichier1.o, fichier2.o et fichier3.o pour obtenir prog.

En pratique, make se rend compte de ce qu'il faut faire grâce aux dates de dernière modification de ces fichiers. Puisqu'on vient de modifier fichier1.c, la date de dernière modification de ce fichier est postérieure à celle de fichier1.o. make le recompile alors en utilisant la commande appropriée (celle qui figure dans le fichier Makefile). Comme prog dépend de fichier1.o et que celui-ci lui est postérieur (on vient de le modifier en recompilant fichier1.c), make effectue
aussi l'édition de liens :

$ make

cc -c fichier1.c

cc -o prog fichier1.o fichier2.o fichier3.o -lm

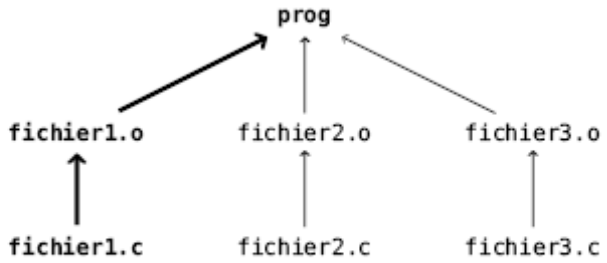


FIGURE 2 – *Principe de fonctionnement de la commande* `make`

Ce mécanisme est illustré de manière plus imagée dans la figure 2. La commande make remonte l'arbre des dépendances depuis les feuilles (parties terminales ne dépendant de rien) jusqu'à la racine (c'est la cible à atteindre). Dès qu'il trouve un fichier dont la date de dernière modification est plus récent que celle du fichier qui suit, il exécute la commande appropriée, puis poursuit sa montée.

Les commentaires

Il est possible (et même recommandé) d'indiquer des commentaires dans un fichier Makefile. Un commentaire commence par un croisillon # et s'étend jusqu'à la fin de la ligne :

 # fichier Makefile permettant de compiler prog

Quelques cibles utiles

Une cible n'est pas toujours un nom de fichier. Cela peut être également une chaîne de caractères quelconque permettant soit la construction de plusieurs dépendances (c'est le cas dela cible all que nous allons aborder au paragraphe a) soit l'exécution de commandes sans condition de dépendance (c'est le cas de la cible clean que nous allons aborder au paragraphe b).
Une cible cible se construit grâce à la commande :

$ make cible

a) La cible all

Notre exemple n'aboutit à la création que d'un seul fichier exécutable, prog. La cible prog est donc la première dans le fichier Makefile. Mais comment faire si l'on doit construire deux fichiers exécutables, prog1 et prog2, puisque seule une des deux cibles correspondantes pourra figurer en première position dans le fichier Makefile ? La cible all permet de résoudre ce problème, en la plaçant en premier dans le fichier Makefile et en la faisant dépendre de prog1 et de prog2 :

all : prog1 prog2

La cible all n'a généralement pas de commande associée car elle n'est utilisée que pour construire plusieurs autres cibles.

b) La cible clean

La cible clean est utilisée pour faire le ménage. Sa fonction est de supprimer tous les fichiers qui peuvent être recréés afin de ne conserver que ceux qui sont indispensables. Dans notre exemple, on peut supprimer le fichier exécutable prog et les fichiers objets *.o. Généralement, on en profite pour supprimer un éventuel fichier core. Pour notre exemple, la cible clean s'écrit donc ainsi :

clean:

⟶!rm -f prog *.o core

La cible clean n'a généralement pas de dépendances.
L'option -f de la commande rm permet d'éviter l'affichage d'un message d'erreur si l'un des fichiers n'existe pas (ce qui est normalement le cas pour le fichier core).

3.6 Environnement graphique X-WINDOWS

3.6.1 Serveur graphique, bureau et gestionnaire de fenêtres ?

- Le serveur graphique est le programme qui permet de passer en *mode graphique* en utilisant les fonctions avancées de la carte graphique. Il gère notamment le clavier, la ou les souris, les polices de caractères, l'écran (résolution, nombre de couleurs,...) et la carte graphique.
- Le gestionnaire de fenêtres est le programme qui, comme son nom l'indique, gère les différentes fenêtres, et il ne fait normalement que ça ! Il existe de très nombreux gestionnaires de fenêtre sous Linux : Enlightenment, AfterStep, Window Maker, etc...
- Le bureau est le programme qui s'occupe d'afficher un menu, une barre de lancement, une barre des tâches, des icones sur le bureau, etc... Il existe également de nombreux bureaux sous Linux, mais les deux plus connus sont Gnome et KDE.

3.6.2 Description

X (et non X-Window) est une interface graphique, qui a été développée au MIT, permettant de créer des applications graphiques fonctionnant sur diverses plate-formes.

X-Window est l'interface graphique des stations UNIX. Elle est en quelque sorte aux systèmes Unix ce que Microsoft Windows est au monde PC (n'allez surtout pas dire X-windows au risque de vous faire massacrer par un fanatique d'UNIX). L'avantage majeur de ce système est l'utilisation d'une interface graphique en remplacement de certaines commandes.

Sous Linux il existe une implémentation libre du système X-Window appelée XFree86, destinée aux systèmes de type Unix. XFree86 supporte un nombre très important de cartes vidéo, mais certaines ne sont pas encore supportées. Toutefois avec la communauté du libre, le portage des pilotes des nouvelles cartes graphiques est de plus en plus rapide !

XFree86 est donc le serveur graphique pour Linux le plus répandu et de loin le plus utilisé. C'est un serveur qui permet d'avoir sur sa propre machine des fenêtres graphiques et éventuellement un bureau comme c'est le cas par exemple avec des postes équipés du système d'exploitation Microsoft Windows. Mais XFree86 fait bien plus que cela. En effet, à la différence d'autres systèmes d'exploitation payants, XFree86 est un serveur graphique. C'est à dire qu'il permet à d'autres personnes qui sont reliées à votre réseau de se connecter à votre machine pour pouvoir exécuter des applications graphiques.

3.6.3 Installation

Nous allons installer le paquet *xserver-xfree86* qui contient le serveur graphique *XFree86* :

apt-get install x-window-system-core

Tableau 1-2. apt-get install x-window-system-core

Nom du paquet	Question	Réponse à choisir
xserver-common	Manage XFree86 4.x server configuration file with debconf? automatically ?	*Yes*
xserver-xfree86	Manage XFree86 4.x server configuration file with debconf?	*No*

3.6.4 Configuration

Pas besoin d'aller mettre les mains dans le cambouis cette fois-ci puisque la configuration va se faire à l'aide de debconf. Pour cela il suffit d'exécuter la commande suivante :

dpkg-reconfigure xserver-xfree86

Tableau 1-3. dpkg-reconfigure xserver-xfree86

Nom du paquet	Question	Réponse à choisir
xserver-xfree86	Manage XFree86 4.x server configuration file with debconf?	*Yes*
xserver-xfree86	Select the desired X server driver	Ce choix va dépendre de votre carte graphique. En général le nom du driver parle de lui-même. Par exemple pour une carte graphique à base de chipset NVidia il faut sélectionner le driver *nv*. Le nom du driver est en rapport avec le constructeur qui fabrique le processeur graphique de votre carte vidéo.
xserver-xfree86	Enter an identifier for your video card	Entrez le nom de votre carte graphique.
xserver-xfree86	Please enter the video card's bus identifier	Ne pas rentrer de valeur ici
xserver-xfree86	Enter the amount of memory (in kB) to be used by your video card	Ne pas rentrer de valeur ici. La quantité de mémoire est detectée automatiquement en général :)
xserver-xfree86	Use kernel framebuffer device interface	*No*
xserver-xfree86	Please select the XKB rule set to use	*xfree86*
xserver-xfree86	Please select your keyboard model	*pc104*
xserver-xfree86	Please select your keyboard layout	*fr*
xserver-xfree86	Please select your keyboard variant	Ne pas rentrer de valeur ici
xserver-xfree86	Please select your keyboard options	Ne pas rentrer de valeur ici

Nom du paquet	Question	Réponse à choisir
xserver-xfree86	Please choose your mouse port	*/dev/psaux*
xserver-xfree86	Please choose the entry that best describes your mouse	*ImPS/2*
xserver-xfree86	Emulate 3 button mouse	*No*. Sauf si vous avez une souris 2 boutons et que vous souhaitez émuler un troisième bouton en cliquant sur les 2 boutons de votre souris en même temps.
xserver-xfree86	Enable scroll events from mouse wheel?	*Yes*. Sauf si vous n'avez pas de molette de défilement sur votre souris.
xserver-xfree86	Enter an identifier for your monitor	Entrez le nom de votre moniteur.
xserver-xfree86	Is your monitor an LCD device?	Ce choix dépend de votre type d'écran. Pour un écran LCD (écran de portable ou écran plat) vous devez répondre *Yes*.
xserver-xfree86	Please choose a method for selecting your monitor characteristics	*Medium*
xserver-xfree86	Please select your monitor(s best video mode	Vous devez choisir la meilleure résolution pour votre moniteur. Référez vous à la documentation qui accompagne ce dernier. Si vous avez plusieurs taux de rafraîchissement possibles, prenez toujours le plus élevé pour un confort visuel optimal.
xserver-xfree86	Select the video modes you would like the X server to use	En général si vous avez bien répondu à la question précédente, vous n'avez rien à changer.
xserver-xfree86	Please select your desired default color depth in bits	*24*
xserver-xfree86	Select the XFree86 server modules that should be loaded by default.	Reportez-vous au tableau ci-dessous pour connaître les modules à charger en fonction de votre carte graphique.
xserver-xfree86	Write default Files section to configuration file?	*Yes*
xserver-xfree86	Write default DRI section to configuration file?	Répondez *Yes* si vous avez chargé le module DRI.

Tableau 1-4. Modules à charger pour le serveur XFree86

Type de carte graphique	dri	glx	GLcore
Driver nVidia propriétaire	*Non*	*Oui*	*Non*
Driver nVidia openSource	*Non*	*Oui*	*Non*
Carte 3D avec DRI/DRM	*Oui*	*Oui*	*Oui*
Autres cartes	*Non*	*Non*	*Non*

Si vous souhaitez reconfigurer votre paquet à l'aide de *debconf*exécutez la commande :

dpkg-reconfigure xserver-xfree86

3.6.5 Utilisation

Une seule commande permet de lancer le serveur X sur votre machine :

$ startx

NB : Si le serveur X ne répond plus, vous pouvez le tuer en maintenant les touches [CTRL], [ALT] et [BACKSPACE] enfoncées. La touche [BACKSPACE] est en fait la touche communément appelée *retour chariot !*

Basculer entre le mode graphique et vos consoles

- Pour aller sur la console n à partir du serveur graphique utilisé la combinaison de touches [CTRL]+[ALT]+[Fn].
- Pour aller de la console m à la console n utilisez la combinaison de touches [ALT]+[Fn].
- Le serveur graphique est par défaut sur la console numéro 7. Soit la combinaison de touches [ALT]+[F7].

Webographie :

1. http://www.labri.fr/perso/preuter/os/os.ppt
2. http://www.commentcamarche.net/contents/pc/bios.php3
3. http://www.ybet.be/oper-95/windows-95-install.htm
4. http://www.commentcamarche.net/contents/dos/dosintro.php3
5. http://www.ccacanada.qc.ca/cours/initiation/
6. http://hautrive.free.fr/winnt/admin/
7. http://homepages.laas.fr/matthieu/cours/unix/unix1.pdf
8. http://www.commentcamarche.net/contents/unix/unix-shell.php3
9. http://www.tuteurs.ens.fr/unix/shell/script.html
10. http://www.babafou.eu.org/ensta/in201/make.pdf
11. http://guide.andesi.org/html/fxfree.html

Autre :

Cours SE.doc Professeur TAYOU DJAMEGNI C