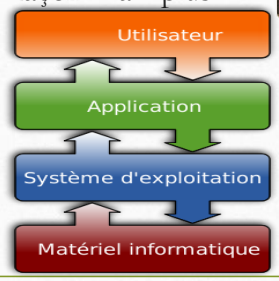
Linux

Pr. EL KABTANE Hamada

Email : youssef.mourdi@ced.uca.ma

Introduction au Système d'exploitation

- Un système d'exploitation est un programme qui doit permettre aux utilisateurs d'utiliser les fonctionnalités d'un ordinateur. Et il doit aussi aider le programmeur à développer des logiciels de la façon la plus efficace possible.
- Le système constitue donc une interface entre l'utilisateur et la machine physique.



Introduction au Système d'exploitation

Malgré les différences de point de vue, de forme, de taille et de type, les ordinateurs se composent de matériel et de logiciels.

- ***Le matériel*** d'un système informatique est composé de processeurs qui exécutent les instructions, de la mémoire centrale qui contient les données et les instructions à exécuter, de la mémoire secondaire qui sauvegarde les informations, et des périphériques d'Entrées/Sorties (clavier, souris, écran, modem, etc.) pour introduire ou récupérer des informations.
- ***Les logiciels*** sont à leur tour divisés en **programmes système** qui font fonctionner l'ordinateur : le **système d'exploitation** et les utilitaires (compilateurs, éditeurs, interpréteurs de commandes, etc.) et en **programmes d'application** qui résolvent des problèmes spécifiques des utilisateurs.

Licence logiciel

La licence est un contrat entre l'éditeur et l'utilisateur

La licence est le contrat qui donne à un utilisateur, sous condition, le droit d'utiliser un produit logiciel. Il pourrait exister autant de licences que de logiciels. Nous allons procéder ici à un classement par grandes familles de licences, en fonction des contraintes qu'elles imposent.

Chaque type de licence apporte un certain nombre d'avantages et d'inconvénients à l'éditeur et à l'utilisateur. Les contrats couvrent généralement quatre aspects :

- ❑ Le droit d'utilisation ;
- ❑ Le droit de redistribution ;
- ❑ Le droit de modification ;
- ❑ La contrepartie financière

Licence logiciel

- **Licence Fixe ou OEM:**

La licence OEM (**Original Equipment Manufacturer**) ou dite fixe est conçue pour être installée sur un ordinateur particulier.

Elle peut utiliser une caractéristique spécifique à cet ordinateur (son adresse MAC par exemple) pour vérifier et contraindre la conformité de l'usage de la licence.

Licence logiciel

- **La licence nominative**

Cette licence est attribuée à un utilisateur particulier, enregistrée sous son nom.

Il peut être en mesure d'installer le programme sur tout ordinateur, mais sera le seul utilisateur agréé à l'utiliser.

Licence logiciel

- **La licence flottante**

La licence flottante fonctionne avec un ordinateur serveur de licences.

Ce dernier décompte le nombre de licences utilisées à un moment précis sur le réseau et va vérifier s'il reste une licence à allouer.

Tant qu'au moins une licence reste disponible, le serveur allouera la licence à n'importe quel ordinateur du réseau la demandant.

Cette licence sera affectée temporairement au poste en question, le temps d'utilisation du logiciel concerné.

Licence logiciel

- **Les partagiciels (sharewares)**

Les partagiciels sont des logiciels propriétaires, disponibles en versions d'évaluation. Si l'utilisateur est satisfait du produit, et qu'il souhaite continuer à l'utiliser, il doit s'acquitter du prix de la licence. Il s'agit plus d'une distinction faite sur le mode de distribution que sur la liberté du logiciel.

Licence logiciel

- **Les logiciels libres**

Les logiciels libres sont un ensemble de logiciels fondés sur un tout autre modèle économique que l'ensemble des logiciels propriétaires que nous avons vus jusqu'à présent. Un logiciel libre est un logiciel dont il est possible de modifier les sources à volonté, et de redistribuer les modifications sans contrainte. Dans cette optique, le logiciel ne doit pas être considéré comme le produit sur lequel la plus-value s'effectue, mais comme un élément d'une stratégie globale.

Licence logiciel

- **Les logiciels libres**

Les licences libres qui couvrent les logiciels libres sont les licences qui donnent au moins les quatre droits suivants aux utilisateurs :

- usage de l'œuvre ;
- étude de l'œuvre pour en comprendre le fonctionnement ou l'adapter à ses besoins ;
- modification (amélioration, extension et transformation) ou incorporation de l'œuvre en une œuvre dérivée ;
- redistribution de l'œuvre et des œuvres dérivées, c'est-à-dire sa diffusion à d'autres usagers, y compris commercialement.

Les fonctions d'un système d'exploitation

Les systèmes d'exploitation modernes sont constitués de centaines de milliers, voire de millions de lignes de code. Ils ont comme fonctions principales :

– Chargement et lancement des programmes.

– Gestion des processeurs, de la mémoire, des périphériques.

– Gestion des processus (programmes en cours d'exécution) et des fichiers.

– Protection contre les erreurs et la détection des erreurs, etc.

FINALITÉS DU SYSTÈME D'EXPLOITATION

- Fournir à l'utilisateur une machine, plus simple à programmer (gestion de matériel de façon transparente).
- Par exemple : donner une interface commune pour écrire sur un fichier qui peut se trouver sur disque avec un système de fichier fat, ou clé USB avec un système de fichier ext3.
- Gestion des ressources matériel

Optimiser l'usage de la machine (taux d'occupation du CPU, minimisation des mouvements des têtes de lecture des disques, gestion de mémoire centrale, gestion de l'énergie sur les systèmes portables, etc.)

LES FONCTIONS D'UN SYSTÈME D'EXPLOITATION

- Les rôles de l'OS sont divers et concernent notamment :

➢ La gestion du processeur : le système d'exploitation est chargé de gérer l'allocation du processeur entre les différents programmes grâce à un algorithme d'ordonnancement.

➢ Gestion de la mémoire vive : le système d'exploitation est chargé de gérer l'espace mémoire alloué à chaque application et à chaque usager. En cas d'insuffisance de mémoire physique, le système d'exploitation peut créer une zone mémoire sur le disque dur, appelée « mémoire virtuelle ».

➢ La mémoire virtuelle permet de faire fonctionner des applications nécessitant plus de mémoire qu'il n'y a de mémoire vive disponible sur le système. En contre partie, cette mémoire est beaucoup plus lente.

LES FONCTIONS D'UN SYSTÈME D'EXPLOITATION

➢ Gestion des entrées/sorties : le système d'exploitation permet d'unifier et de contrôler l'accès des programmes aux ressources matérielle par l'intermédiaire des pilotes.

➢ Gestion de l'exécution des applications : le système d'exploitation est chargé de la bonne exécution des applications en leur affectant les ressources nécessaires à leur bon fonctionnement. Il permet à ce titre de « tuer » une application ne répondant plus correctement.

➢ Gestion des droits : le système d'exploitation est chargé de la sécurité liée à l'exécution des programmes en garantissant que les ressources ne sont pas utilisées que par des programmes et utilisateur possédant les droits adéquats.

LES FONCTIONS D'UN SYSTÈME D'EXPLOITATION

- Gestion des fichiers : le système d'exploitation gère la lecture et l'écriture dans les fichiers et les droits d'accès aux fichiers par les utilisateurs et les applications.
- Gestion des informations : le système d'exploitation fournit un certain nombre d'indicateurs permettant de diagnostiquer le bon fonctionnement de la machine.
- Gestion des utilisateurs: gérer les différents utilisateurs et leurs comptes et données.

CLASSIFICATIONS DES SYSTÈMES D'EXPLOITATION

- **Mono \ multitâche**
  - **Le système mono-tâche**

Il ne gère qu'une seule tâche à la fois (un seul programme). Quand le programme est lancé, il utilise seul les ressources de la machine et ne rend la main au système d'exploitation qu'en fin d'exécution, ou en cas d'erreur.

Exemple : MS-DOS.

  - **Le système multitâche**

Il gère simultanément plusieurs programmes sur une même machine. Il permet de partager le temps du processeur pour plusieurs programmes, ainsi ceux-ci sembleront s'exécuter simultanément. Le principe est d'allouer du temps à différents programmes (taches ou processus) fonctionnant simultanément. Ces tâches seront tour à tour actives, en attente, suspendues ou détruites, suivant l'algorithme qui leur est associée. Le temps alloué peut être fixe ou variable suivant le type de partage géré par le SE.

Exemple : Windows 95, 98, Xp, Vista, Ubuntu, Suse, Debian, ...

CLASSIFICATIONS DES SYSTÈMES D'EXPLOITATION

- **Mono \ multi-utilisateur**
  - **Le système mono-utilisateur**

Un système d'exploitation mono-utilisateur, est un système qui a été développé et conçu pour un seul utilisateur à la fois. On peut trouver un système mono-utilisateur mono-tâche ou multitâches.

Exemple : MS-DOS, Unix.

  - **Le système multi-utilisateur**

Un système d'exploitation **multi-utilisateur** est conçu pour permettre à plusieurs utilisateurs d'utiliser l'ordinateur simultanément, tout en limitant les droits d'accès de chacun afin de garantir l'intégrité de leurs données.

- Gestion d'environnement propre à chaque utilisateur(identification, ressources propres)
- Sécurité d'accès aux programmes et aux données
- Notion de droits d'accès

Exemple : Windows 95, 98, Xp, Vista,Ubuntu, Suse, Debian, ...

CLASSIFICATIONS DES SYSTÈMES D'EXPLOITATION

- **Selon leur architecture**
  - **Système centralisés** : l'ensemble du système est entièrement présent sur la machine considérée. Les machines éventuellement reliées sont vues comme des entités étrangères disposant elle aussi d'un système centralisé. Le système ne gère que les ressources de la machine sur laquelle il est présent.
  - **Système répartis** (« destributed systems ») : les différentes abstractions du système sont réparties sur un ensemble (domaine) de machines (site). Le système d'exploitation réparti apparait aux yeux de ses utilisateurs comme une machine virtuelle monoprocesseur même lorsque cela n'est pas le cas. Avec un système réparti, l'utilisateur n'a pas à se soucier de la localisation des ressources. Quand il lance un programme, il n'a pas à connaitre la machine de domaine qui l'exécutera. Ils offrent des solutions aux problèmes de la résistance aux pannes.
  - Exemple Plan 9 , Inferno, DPCX

CLASSIFICATIONS DES SYSTÈMES D'EXPLOITATION

➢ **Préemptif / Coopératif**

- Un système d'exploitation multitâche est capable d'exécuter plusieurs programmes simultanément. Il existe deux types de fonctionnement multitâche : le multitâche préemptif et le multitâche coopératif, le premier étant globalement le plus rationnel et le plus efficace. En multitâche préemptif, un module du système d'exploitation se charge de partager de façon équilibrée le temps de calcul entre les différents programmes actifs. Une notion de priorité lui permet de hiérarchiser les programmes. En multitâche coopératif, il revient aux applications actives de se répartir elles-mêmes le temps de calcul. Il n'existe alors aucune hiérarchie entre les différentes applications.

CLASSIFICATIONS DES SYSTÈMES D'EXPLOITATION

➢ **Préemptif / Coopératif**

- Les systèmes d'exploitation **préemptifs** gèrent le temps processeur alloué à chaque application. Un commutateur de tâches intervient pour répartir l'allocation des ressources. Des degrés de priorité sont accordés à chaque application. Chaque application peut être interrompus sans interagir avec les autres applications. le noyau garde toujours le contrôle (qui fait quoi, quand et comment).
- Les systèmes d'exploitation **coopératifs**: Une seule application peut monopoliser toutes les ressources de l'ordinateur, et ne rendre la main aux autre application uniquement quand elle aura terminer.
- Un systèmes coopératif permet à plusieurs applications de fonctionner et d'occuper des plages mémoire, laissant le soin à ces applications de gérer cette occupation, ce qui risque de bloquer tout le système.

CLASSIFICATIONS DES SYSTÈMES D'EXPLOITATION

➢ **Les système d'exploitation temps réel**

Un système d'exploitation temps réel est un système d'exploitation multitâche destiné aux applications temps réel. Ces applications comprennent les systèmes embarqués, des robots industriels, les vaisseaux spatiaux, les systèmes de contrôle commande industriel, le matériel de recherche scientifique, etc.

Exemple de SE temps réel : Adeos, ART Linux, ERIKA Enterprise, LynxOS, pSOS,etc.

CLASSIFICATIONS DES SYSTÈMES D'EXPLOITATION

- Par interfaces :
  - La ligne de commande, **le mode texte** avec le clavier.
  - L'interface graphique (GUI pour Graphical User Interface), **le mode graphique**, avec un pointeur comme une souris.
- Par nombre d'application qui tournent en simultané :
  - Les systèmes d'exploitation **mono tâche.**
  - Les systèmes d'exploitation **multi tâches**
  - Les systèmes d'exploitation **préemptifs**
  - Les systèmes d'exploitation **coopératifs**.
- Par nombre d'utilisateurs :
  - Les systèmes d'exploitation **mono utilisateurs**.
  - Les systèmes d'exploitation **multi utilisateurs** Par connectivité réseau :
  - Les systèmes d'exploitation **clients**.
  - Les systèmes d'exploitation **serveurs**.
- Par nombre de bits des instructions des programmes qui sont développés pour fonctionner avec tel ou tel système :
  - Les systèmes d'exploitation **16 bits**
  - Les systèmes d'exploitation **32 bits**
  - Les systèmes d'exploitation **64 bits**
- Par nombre de processeur :
  - Les systèmes d'exploitation **mono processeur**
  - Les systèmes d'exploitation **multi processeur**

Interactions utilisateur/système

Pour un utilisateur, le système d'exploitation apparaît comme un ensemble de procédures, trop complexes pour qu'il les écrive lui-même. Les bibliothèques des **appels système** sont alors des procédures mises à la disposition des programmeurs. Ainsi un programme C/C++ peut utiliser des appels système d'Unix/Linux comme open(), write() et read() pour effectuer des Entrées/Sorties de bas niveau.

L' **interpréteur de commandes** constitue une interface utilisateur/système. Il est disponible dans tous les systèmes. Il est lancé dès la connexion au système et invite l'utilisateur à introduire une commande. L'**interpréteur de commandes** récupère puis exécute la commande par combinaison d'appels système et d'outils (compilateurs, éditeurs de lien, etc.). Il affiche les résultats ou les erreurs, puis se met en attente de la commande suivante.

Interactions utilisateur/système

L'introduction du graphisme dans les interfaces utilisateur a révolutionné le monde de l'informatique. L'interface graphique a été rendue populaire par le Macintosh de Apple. Elle est maintenant proposée pour la plupart des machines.

Appels système

En général, les processeurs ont deux modes de fonctionnement :

- – Le **mode superviseur** (noyau, privilégié ou maître) : pour le système d'exploitation, où toutes les instructions sont autorisées.
- – Le **mode utilisateur** (esclave) : pour les programmes des utilisateurs et les utilitaires, où certaines instructions ne sont pas permises.

Ces modes de fonctionnement assurent la protection du système d'exploitation contre les intrusions et les erreurs. Ce n'est pas le cas des systèmes mono-utilisateur comme DOS qui a un seul mode de fonctionnement : le mode noyau. Ils ne est pas protégé et donc peu fiable.

Appels système

Un **appel système** a pour but : changer de mode d'exécution pour passer du mode utilisateur au mode maître, récupérer les paramètres et vérifier la validité de l'appel, de lancer l'exécution de la fonction demandée, de récupérer la (les) valeur(s) de retour et de retourner au programme appelant avec retour au mode utilisateur.

Appels système

- **Les fichiers**

Un **fichier** est un ensemble de données enregistrées de façon à être lues et traitées par ordinateur. Les fichiers peuvent être regroupés dans des répertoires. Un **répertoire** peut contenir soit des fichiers, soit d'autres répertoires dans une structure arborescente.

L'accès aux fichiers se fait en spécifiant un **chemin d'accès**, c'est-à dire la liste des répertoires à traverser pour accéder au fichier. Un chemin d'accès est **absolu** si le point de départ est le répertoire racine. Un chemin d'accès est **relatif** si le point de départ est le répertoire courant. Les appels système permettent de créer les fichiers et les répertoires, ainsi que de les supprimer, de les ouvrir, de les lire et de les modifier.

Appels système

- **Les processus**

Un **processus** est un programme en cours d'exécution. Il est composé d'un **programme exécutable** (code), un **compteur ordinal** (**co**), un ensemble de données, une pile d'exécution et autres registres et informations nécessaires à l'exécution.

Les appels systèmes permettent notamment la création et l'arrêt des processus. Un processus peut créer un ou plusieurs processus fils qui, à leur tour, peuvent créer des processus fils dans une structure arborescente. Les processus peuvent se synchroniser et communiquer entre eux. Actuellement, on utilise de plus en plus le concept de **processus légers.**

Les processus légers sont un moyen de raffiner et de diviser le travail normalement associé à un processus.

Exemples de systèmes d'exploitation

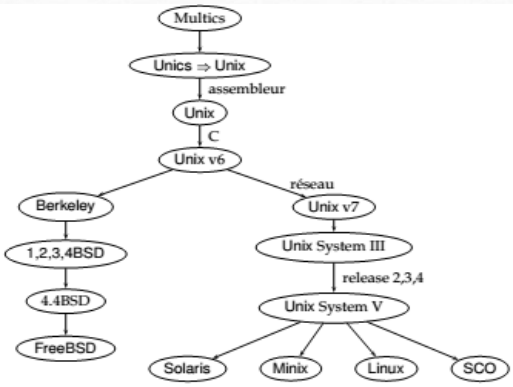
- Il y a plusieurs types de systèmes d'exploitation actuellement :
  – MS-DOS, Windows
  – OS/2, Mac-OS
  – Unix (AIX, Xenix, Ultrix, Solaris, etc.)
  – Linux

Unix

- Le système Unics (Uniplexed Information and Computing Service) a été créé aux laboratoires AT&T de Bell, en 1969 par Ken Thompson, et modifié et baptisé par Brian Kernighan. Il a été une version réduite de **Multics** (Multiplexed Information and Computing Service). Peu après on a changé le nom Unics par Unix, et à partir de ce moment le système Unix a commencé un long chemin de développement technique.

Historique Unix

- Le système Unix a connu un véritable succès, lorsqu'il fut récrit en langage C en 1973 par Denis Ritchie et Thompson. L'Université de Californie à Berkeley a obtenu une copie de la version 6 du système Unix. AT&T et Berkeley ont séparément apporté de nombreuses modifications et améliorations au système Unix (**SystemV** d'AT&T et **4.4BSD** de Bekerley).



Historique Unix

Le projet Posix (Portable Operating System UnIX) de normalisation du système Unix a permis de développer un standard qui définit un ensemble de procédures. Tout système Unix conforme à Posix fournit ces procédures ou appels système standards. Ces procédures constituent la bibliothèque standard d'Unix. Tout logiciel écrit en n'utilisant uniquement les procédures de la norme Posix devrait fonctionner sur tous les systèmes Unix conformes à cette norme. Une version gratuite d'Unix porte le nom de Linux (code source disponible). Elle a été créée par Linus Torvalds en 1991. Par la suite, un grand nombre de programmeurs ont contribué à son développement accéléré.

Historique Unix

FreeBSD est une autre version gratuite d'Unix. Il est un système d'exploitation pour x86 compatibles, DEC Alpha, et architectures PC-98. Il est dérivé de BSD Unix et développé par une grande communauté d'individus (http ://www.freebsd.org).

Les distributions Linux

- **Slackware**

La distribution **Slackware** est l'une des plus anciennes distributions. Elle était même livrée sur disquette. Durant les toutes premières années la Slackware était la distribution de référence pour apprendre à utiliser Linux. Elle est extrêmement dépouillée. Son installateur est réduit à la plus simple expression et la plupart de la configuration doit être effectuée à la main.

Les distributions Linux

- **Debian**

Le projet Debian a été fondé en 1993 par Ian Murdock à une époque où l'idée même de distribution Linux en était encore à ses commencements. Le nom Debian provient de Debra (la femme de Murdock) et Ian. Debian a longtemps été la seule distribution entièrement et uniquement composée de logiciels libres et Open Source ce qui lui vaut toujours le nom officiel de Debian GNU/Linux.

Les distributions Linux

- **Ubuntu**

Le milliardaire sudafricain Mark Shuttleworth, principalement connu du monde entier pour avoir été l'un des premiers touristes de l'espace, mais aussi des informaticiens pour avoir fait fortune en revendant sa société Thawte spécialisée dans la sécurité à Verisign, est un vrai informaticien qui a contribué au projet Debian. Devant les quelques inconvénients de la distribution il crée la distribution Ubuntu Linux en 2005 avec un budget initial de 10 millions de dollars pour rémunérer les développeurs. Le mot Ubuntu est un mot du langage africain bantou signifiant « humanité aux autres » ou encore « je suis ce que je suis grâce à ce que nous sommes tous ». Cette définition reflète ce qu'est la distribution : un dérivé de Debian dont le but est de fournir des logiciels plus récents et très fortement axés sur la convivialité et l'ergonomie à l'aide du support du plus grand nombre.

Les distributions Linux

- **Red Hat et Fedora**

La société Red Hat fondée en 1995 par Robert Young et Marc Ewing, elle édite la célèbre distribution éponyme dont la première version officielle date de 1994 (la société a été fondée après la sortie de la distribution). Le système de package RPM est apparu avec la version 2.0. Les distributions Red Hat ont très fortement marqué les esprits car elles sont restées la référence pendant presque dix ans. Chaque version était innovante tant dans l'intégration des logiciels que dans son installateur (appelé anaconda) et ses outils de configuration.

Cependant en 2003 la version 9.0 est la dernière destinée officiellement au grand public. Les versions suivantes ont été confiées au projet communautaire **Fedora** qui continue tous les six mois à sortir une nouvelle version. Red Hat se concentre maintenant sur le monde de l'entreprise avec des distributions commerciales appelées **RHEL** (*Red Hat Enterprise Linux*)

Les distributions Linux

- **Red Hat et Fedora**

Le projet Fedora, il suit un cycle de développement rapide et reste destiné au grand public. Son installation est simple. Cependant l'ensemble manque un peu de cohérence (par exemple l'outil de partitionnement des disques n'est accessible que durant l'installation) ce qui en fait une distribution idéale pour tous ceux qui, amateurs éclairés, souhaitent rentrer un peu plus dans le détail.

Les distributions Linux

- **Mandriva (exMandrake)**

Mandriva Linux (exMandrake) est une distribution dérivée et longtemps entièrement compatible avec la distribution Red Hat. Elle a été créée par Gaël Duval afin d'intégrer à la distribution l'environnement de bureau graphique KDE contrairement à Red Hat qui intégrait l'environnement GNOME. Pendant plusieurs années Mandrake a été la distribution phare en forte compétition avec Red Hat. Mandrake était en effet (et est toujours) plus conviviale. Son processus d'installation est un modèle du genre et son utilisation des plus simples. Renommée Mandriva suite au rachat de la société Connectiva, la distribution est pourtant en perte d'audience depuis quelques temps.

Les distributions Linux

- **OpenSUSE**

OpenSUSE est une distribution d'origine allemande datant de 1992. Le nom de l'entreprise lui même était un hommage au célèbre **Konrad Zuse** l'inventeur des ordinateurs modernes.

La distribution est originellement basée sur la distribution Slackware. En 1996 SuSE se rapproche d'une distribution française appelée **Jurix** créée par Florian La Roche qui est utilisée comme base à la place de Slackware. Cette même année le développement de l'outil YaST est démarré et la version 4.2, en fait totalement nouvelle, sort. Au même moment SuSE utilise le nouveau gestionnaire de packages de Red Hat appelé RPM.

Début 1997 SuSE tente l'aventure américaine en installant de nouveaux bureaux à Oakland. Entre 1997 et 2003 la distribution SuSE ne cesse d'être améliorée pour devenir une référence en matière de simplicité d'installation, d'administration et d'utilisation.

Propriétés d'UNIX

- Lunix est un système d'exploitation :
  - Multiutilisateurs, multitâches, cela signifie que plusieurs utilisateurs peuvent accéder simultanément au système et exécuter un ou plusieurs programmes.
  - Orienté temps partagé, c'est-à-dire que les ressources du processeur et du système sont réparties entre les utilisateurs.
  - Dont les systèmes de fichiers sont hiérarchisés en arbre, plusieurs systèmes de fichiers peuvent être rattachés au système de fichiers principal ; chaque système de fichiers possède ses propres répertoires, qui offre une compatibilité totale des entrées-sorties (pour Unix, les périphériques sont des fichiers), ce qui rend le système indépendant du matériel et en assure la portabilité ;

Propriétés d'UNIX

- Gestion de la mémoire virtuelle : un mécanisme d'échange entre la RAM et le disque dur permet de pallier un manque de RAM et optimise le système.
- Processus réentrants : les processus exécutant le même programme utilisent une seule copie de celui-ci en RAM.
- Interface utilisateur interactive (shell) : elle est constituée d'un programme séparé du noyau permettant à l'utilisateur de choisir son environnement de travail. Elle intègre un langage de commandes très sophistiqué (shell scripts).

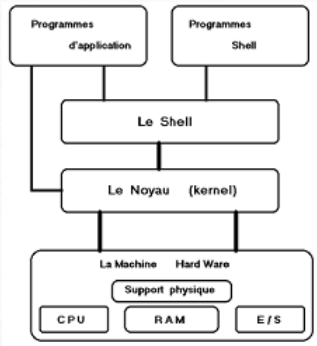
Structure d'Unix

- ***Le Noyau***

Le noyau est le programme qui assure la gestion de la mémoire, le partage du processeur entre les différentes tâches à exécuter et les entrées/sorties de bas niveau. Il est lancé au démarrage du système (le boot) et s'exécute jusqu'à son arrêt.

C'est un programme relativement petit, qui est chargé en mémoire principale.

Le rôle principal du noyau est d'assurer une bonne répartition des ressources de l'ordinateur (mémoire, processeur(s), espace disque, imprimante(s), accès réseaux, …) sans intervention des utilisateurs. Il s'exécute en mode ***superviseur,*** c'est à dire qu'il a accès à toutes les fonctionnalités de la machine : accès à toute la mémoire, et à tous les disques connectés, manipulations des interruptions, etc.

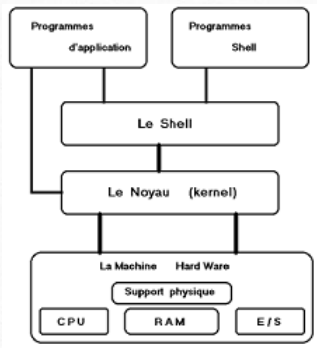


Structure d'Unix

- ***Le Noyau***

Tous les autres programmes qui s'exécutent sur la machine fonctionnent en **mode *utilisateur* :** il leur est interdit d'accéder directement au matériel et d'utiliser certaines instructions. Chaque programme utilisateur n'a ainsi accès qu'à une certaine partie de la mémoire principale.

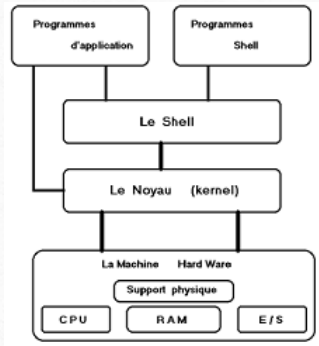
Lorsque l'un de ces programmes désire accéder à une ressource gérée par le noyau, par exemple pour effectuer une opération d'entrée/sortie, il exécute un appel système. Le noyau exécute alors la fonction correspondante, après avoir vérifié que le programme appelant est autorisé à la réaliser.



Structure d'Unix

- **Shell**

Le shell est l'interpréteur de commandes. Quand un utilisateur tape des commandes Unix, ces commandes sont lues par le shell qui effectue éventuellement des traitements avant de lancer l'exécution de la commande. Le shell est une couche logicielle bien séparée du noyau. Il existe plusieurs shells dont les plus utilisés sont :



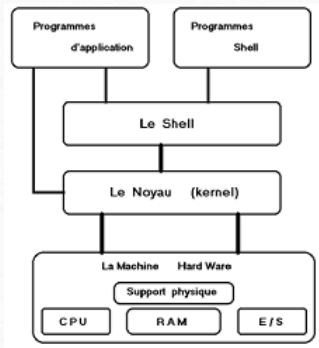
Structure d'Unix

- **Shell**

- le Bourne shell sh, le shell standard d'Unix AT&T,

- le C-shell csh, le shell d'Unix BSD ; sa syntaxe rappelle le langage C,

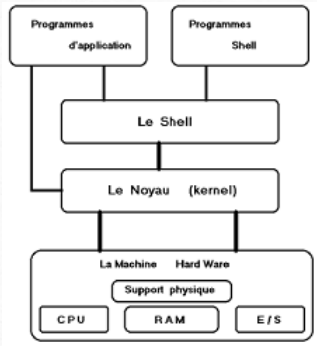
- le Korn-shell ksh est une extension du Bourne shell. Il possède toutes les commandes du Bourne shell (et il se comporte presque exactement comme lui pour ces commandes) et il comprend aussi d'autres commandes et fonctionnalités qui facilitent le travail de l'utilisateur comme, par exemple, la gestion de l'historique des commandes tapées par l'utilisateur, qui existe aussi dans le C-shell. On le trouve maintenant dans la plupart des distributions Unix.



Structure d'Unix

- **Shell**

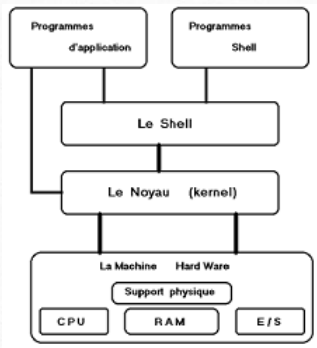
Certaines commandes à la disposition de l'utilisateur, sont programmées dans le shell et celui-ci peut donc les exécuter directement. Elles sont peu nombreuses ; on trouve par exemple les commandes **cd** ou **pwd**. On les appelle les commandes internes au shell. Les autres commandes sont des commandes externes au shell. Pour les exécuter le shell lance un programme qui correspond à un fichier exécutable situé dans l'arborescence des fichiers.



Structure d'Unix

- **Shell**

Le shell possède un véritable langage avec des structures de programmation (alternatives, répétitions,...) et l'utilisateur peut écrire ses propres commandes dans ce langage (le programme s'appelle un shell script). Une fois écrites, ces nouvelles commandes peuvent être utilisées exactement comme les commandes classiques d'Unix.



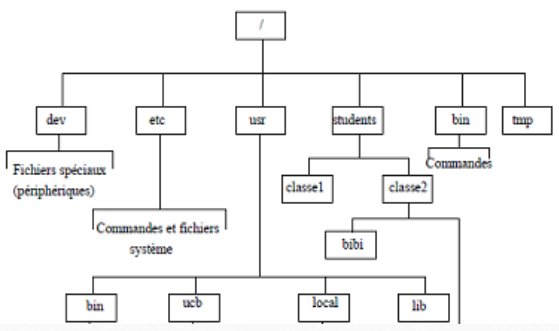
Organisation des systèmes de fichiers

- **Système de fichiers**

Les fichiers d'Unix sont enregistrés dans un ensemble structuré hiérarchiquement en arbre, appelé système de fichiers. Un système de fichiers est composé d'une racine et de nœuds qui sont des fichiers répertoires (ces fichiers contiennent des références à d'autres fichiers), et de fichiers ordinaires qui contiennent des données et des programmes.

Organisation des systèmes de fichiers

En général, plusieurs systèmes de fichiers sont "montés" sur le système "racine", c'est à dire celui qui contient le répertoire « / ». Ces systèmes sont enregistrés sur des disques physiques différents ou sur un même disque mais sur des partitions différentes (un disque physique peut être divisé en plusieurs disques logiques appelés partitions). Tous ces systèmes de fichiers sont vus par l'utilisateur comme un seul système.



Organisation des systèmes de fichiers

❖ Structure du système:

Répertoire	Signification
/	Répertoire racine. Point de départ de toute la hiérarchie du système de fichiers.
/dev/	Répertoire contenant tous les fichiers spéciaux permettant d'accéder aux périphériques.
/etc/	Répertoire contenant tous les fichiers de configuration du système.
/tmp/	Répertoire permettant de stocker des données temporaires.
/home/	Répertoire contenant les répertoires personnels des utilisateurs.
/lib/	Répertoire contenant les bibliothèques partagées

Organisation des systèmes de fichiers

- **Types de fichiers**

Unix utilise les types de fichiers suivants :

Fichiers ordinaires : ils contiennent les données ou les programmes. Ils n'ont aucune structure particulière : ils sont considérés comme une suite d'octets.

On peut distinguer trois grands types de fichiers ordinaires :

- Les fichiers binaires exécutables, écrits dans le code du processeur de la machine, qui ont une structure particulière reconnue par le noyau Unix.
- Les fichiers de textes qui sont structurés en lignes.
- Les autres fichiers qui n'ont pas de structure particulière pour Unix (mais peuvent avoir une structure particulière adaptée au logiciel qui les a créés, les fichiers base de données par exemple).

Organisation des systèmes de fichiers

- **Types de fichiers**

Répertoires : ils sont les nœuds de la structure arborescente des fichiers. Ils contiennent les identificateurs (i-nodes) d'autres fichiers. Ils correspondent à des sous-bibliothèques ou dossiers qui contiennent d'autres fichiers.

Fichiers spéciaux : ils sont liés à un périphérique ; ils correspondent à des programmes (pilotes ; drivers en anglais) qui gèrent les échanges avec les périphériques : disques, terminaux, imprimantes, lecteurs de bandes, etc.

Sockets : ils sont utilisés pour les liaisons interprocessus (entre les processus).

Nomenclature des fichiers

On ne peut pas donner n'importe quel nom à un fichier, il faut pour cela suivre quelques règles simples. Ces règles sont valables pour tous les types de fichiers.

Sur les anciens systèmes Unix un nom de fichier ne pouvait pas dépasser 14 caractères. Sur les systèmes actuels, dont Linux, on peut aller jusqu'à 255 caractères.

Nomenclature des fichiers

Un point extrêmement important : Linux fait la distinction entre les noms de fichiers en minuscules et en majuscules.

Toto, TOTO, ToTo et toto sont des noms de fichiers différents, avec un contenu différent,

La plupart des caractères (les chiffres, les lettres, les majuscules, les minuscules, certains signes, les caractères accentués) sont acceptés, y compris l'espace. Cependant quelques caractères sont à éviter car ils ont une signification particulière au sein du shell : & ; ( ) ~ <espace> \ / | ` ?

Les chemins

Les chemins permettent de définir un emplacement au sein du système de fichiers. C'est la liste des répertoires et sous-répertoires empruntés pour accéder à un endroit donné de l'arborescence jusqu'à la position souhaitée (répertoire, fichier). Un nom de fichier est ainsi généralement complété par son chemin d'accès. C'est ce qui fait que le fichier toto du répertoire rep1 est différent du fichier toto du répertoire rep2.

Les chemins

- **Un chemin absolu:**

Le nom de chemin ou path name d'un fichier est la concaténation, depuis la racine, de tous les répertoires qu'il est nécessaire de traverser pour y accéder, chacun étant séparé par le caractère /. C'est un chemin absolu comme celui- ci. /home/toto/Docs/Backup/fic.bak

Un chemin absolu ou complet :

- démarre de la racine, donc commence par un /,
- décrit tous les répertoires à traverser pour accéder à l'endroit voulu,
- ne contient pas de . ni de ..

Les chemins

- **Un chemin relatif :**

Un nom de chemin peut aussi être relatif à sa position courante dans le répertoire. Le système (ou le shell) mémorise la position actuelle d'un utilisateur dans le système de fichier, le répertoire actif. Vous pouvez accéder à un autre répertoire de l'arborescence depuis l'emplacement actuel sans taper le chemin complet uniquement en précisant le chemin le plus court relativement à votre position actuelle au sein de l'arborescence.

Il faut pour cela souvent utiliser deux entrées particulières de répertoires :

- Le point . représente le répertoire courant, actif. Il est généralement implicite.
- Les doubles points .. représentent le répertoire de niveau inférieur.

Les chemins

- **Un chemin relatif :**

Un chemin relatif :

- décrit un chemin relatif à une position donnée dans l'arborescence, généralement (mais pas toujours) depuis la position courante ;
- décrit en principe le plus court chemin pour aller d'un point à un autre ;
- peut contenir des points ou des doubles points ;

Les chemins

- **Un chemin relatif :**

- Documents/Photos est un chemin relatif : le répertoire Documents est considéré comme existant dans le répertoire courant ;
- ./Documents/Photos est un chemin relatif parfaitement identique au précédent, sauf que le répertoire actif (courant) est explicitement indiqué par le point. ./Documents » indique explicitement le répertoire Documents dans le répertoire actif ;
- /usr/local/../bin est un chemin relatif : les .. sont relatifs à /usr/local et descendent d'un niveau vers /usr. Le chemin final est donc /usr/bin.

Connexions & Déconnexions

❖ Connexion logique:

La connexion logique doit être établie entre un terminal (Ecran+ Clavier) et l'ordinateur sur lequel on va travailler. Généralement il suffit d'allumer la machine.

❖ Initiation de la session de travail:

Cette étape consiste à se s'identifier auprès du système:

- Login : il faut rentrer le nom d'utilisateur
- Password : il faut saisir son mot de passe. Il est généralement chiffré lors de la saisie.
- L'utilisateur est effectivement prêt à travailler quand il reçoit l'invite du système consistant d'un marqueur au début de ligne. Ce marqueur est variable selon les machines (ex : $ ou nom_utilisateur@nom_machine>)
- Le caractère de terminaison peut avoir d'autres significations :
  - un $ indique que l'utilisateur n'a pas de pouvoirs particuliers, comme le >.
  - un # indique que l'utilisateur est l'administrateur root qui a tous les pouvoirs.

Connexions & Déconnexions

❖ Déconnexion :

- Si on est sur un environnement pour se déconnecter, il suffit d'aller au menu « logout ».
- Sur un environnement de lignes de commande, la commande « exit ou logout » suffit pour terminer la session.

Répertoire personnel

- Répertoire Personnel ↔ Répertoire de Connexion ↔ Home Directory
- À chaque utilisateur connu du système est associé un répertoire de connexion. L'utilisateur y place ses fichiers personnels, et peut y créer autant de sous-répertoires qu'il le désire.
- **Exemple :** Soit le répertoire de connexion : ***/home/user***
  - Après le login, l'interpréteur de commande a pour répertoire courant le répertoire de connexion de l'utilisateur.
  - Le répertoire de connexion contient aussi certains fichiers de configuration permettant à l'utilisateur de personnaliser son environnement de travail. Ces fichiers sont normalement invisibles (car leur nom commence par un point, *voir la commande ls*).
- À tout moment, on peut revenir au répertoire de connexion grâce à la commande cd (en tapant tout simplement cd (sans options) : $ cd
- On peut aussi spécifier un chemin à partir du répertoire de connexion d'un utilisateur en utilisant le caractère ~. Par exemple, ~/fichier1 désigne le fichier /home/user/fichier1

Les commandes - Utiliser le shell-

- Dans le terminal, le clavier s'utilise comme d'habitude. Vous pouvez vous déplacer sur la ligne avec les flèches de droite et de gauche du clavier et effacer des caractères avec les touches [Retour arrière] et [Suppr]. Vous lancez l'exécution de la commande que vous avez saisi en appuyant sur la touche [Entrée].
  - La commande **date** indique la date et l'heure actuelles.

```
ubuntu@ubuntu:~$ date
jeudi 11 octobre 2018, 23:59:10 (UTC+0200)
```

  - Une commande pratique, **pwd**, permet de savoir à quel endroit vous vous situez dans les répertoires.

```
ubuntu@ubuntu:~$ pwd
/home/ubuntu
```

Les commandes - Utiliser le shell-

- Méta caractère du shell
  - Caractères ➔ Signification
    - tabulation, espace ➔ appelés white characters ;
    - retour chariot ➔ fin de la commande à exécuter
    - ; ➔ séparateur de commandes sur une seule ligne
    - ( ) ➔ exécution des commandes dans un sous-shell
    - { } ➔ exécution des commandes en série
    - ' " \ ➔ appelés quote characters ; changent la façon dont le shell interprète les caractères spéciaux
    - < > << >> ` | ➔ caractères de redirection d'entrées/sorties (E/S)
    - * ? [ ] [^] ➔ caractères de substitution de noms de fichiers
    - $ ➔ valeur d'une variable

Les commandes

Interprétation des commandes par le shell

Lorsque l'utilisateur saisit une commande dans le terminal texte, par exemple la commande ***allo***, l'interpréteur de commandes suit la procédure suivante pour l'exécuter :

- (1) Si allo est l'une de ses commandes internes, il l'exécute puis redonne la main à l'utilisateur (en réaffichant l'invite de commandes).
- Sinon, il passe à l'étape suivante.

❖**Interprétation des commandes par le shell**

Il parcourt le contenu d'une variable d'environnement appelée **PATH**. Le contenu de **PATH** est une liste de répertoires dans lesquels il faut chercher les binaires correspondant à la commande à exécuter. Par exemple, si **PATH** contient les répertoiressuivants (dans l'ordre) :

/usr/bin

/bin

/home/user/bin

alors l'interpréteur de commandes va chercher les commandes :

/usr/bin/allo

/bin/allo

/home/user/bin/allo

```
guest-0vzido@ubuntu:~$ allo
No command 'allo' found, did you mean:
 Command 'aldo' from package 'aldo' (universe)
allo: command not found
```

(3) Si aucun de ces répertoires ne contient une commandes allo, l'interpréteur de commandes affiche un message d'erreur pour indiquer à l'utilisateur qu'il ne connaît pas cette commande.

Les commandes

❖ **Syntaxe des commandes:**

- Les commandes sont exprimés ainsi :

  nom_commande [options] [arguments]

- Le caractère séparateur entre les commandes c'est blanc exprimé par un ESPACE.
- Les **options** commencent par le caractère – (signe moins) suivi de plusieurs lettre clés . Ces options modifient l'exécution de la commande.
- Les **arguments** spécifient les objets (fichiers ou variables) sur lesquels la commande va s'appliquer.
- Une commande peut avoir ni paramètres, ni arguments. Dans ce cas elle exécute l'action par défaut pour laquelle elle est programmée, ou affiche un message d'erreur si ceux-ci sont nécessaires.

Les commandes

- Exemple

Prenant l'exemple de la commande **cal**. Celle-ci admet plusieurs paramètres et arguments. Appelée seule, elle affiche le calendrier du mois en cours et surligne le jour actuel.

La commande **cal** admet deux arguments optionnels. Si un seul est précisé, il s'agit de l'année, et l'intégralité du calendrier de cette année est affichée. Si deux arguments sont précisés, le premier est le mois, le second l'année.

Les commandes

- Exemple

La commande admet aussi quelques paramètres. Vous remarquez que par défaut l'affichage est prévu pour les anglosaxons : la première colonne est un dimanche, représentant le premier jour de la semaine. En France c'est le lundi. Le paramètre -m permet de le préciser. Exemple: **ncal -M 12 1975**

Un second paramètre -3 permet d'afficher les mois précédant et suivant le mois précisé (ou le mois en cours). Exemple: **cal -3 12 1975**

Les commandes

- **Chaîner les commandes**

Vous pouvez exécuter plusieurs commandes sur une seule ligne, les unes après les autres. Pour cela il suffit de les séparer avec un point-virgule.

Exemple: **date;pwd;cal**

Les commandes

- **Commandes internes et externes**

Il existe deux types de commandes :

- Les commandes externes sont des programmes binaires présents en tant que fichiers sur le disque dur (ou tout autre support de données). Quand vous exécutez la commande, ce fichier est chargé en mémoire et lancé en tant que processus.
- Les commandes internes sont internes au shell et exécutées au sein de celui-ci. Ces commandes font partie du programme shell, le bash. Les commandes **cd** ou **pwd** sont deux exemples de commandes internes. Quand vous les exécutez, le shell exécute les fonctions définies en son sein correspondant à celles-ci.

Les commandes

- **Commandes internes et externes**

Vous pouvez distinguer une commande interne d'une commande externe à l'aide de la commande interne **type**. C'est ainsi que **date** est une commande externe, vous constatez que c'est un fichier présent dans **/bin**, tandis que **pwd** est interne au shell.

$ type date

date is hashed (/bin/date)

$ type pwd

pwd is a shell builtin

Les commandes

- **Commandes internes et externes**

Vous pouvez tomber sur certains autres types comme les alias de commandes qui sont des raccourcis de commandes propres au shell. Ainsi le shell bash de certaines distributions Linux proposent des alias comme ll qui correspond en fait à ls-alF.

$ type ll

ll is aliased to `ls -alF'

Les commandes

- **Rappel de l'historique**

Vous trouverez très utile de pouvoir rappeler une commande que vous avez déjà exécutée en naviguant dans l'historique des commandes avec les touches [Flèche en haut] et [Flèche en bas]. La flèche du haut remonte dans l'historique. Si vous avez saisi les deux précédentes commandes (**date** puis **pwd**), le premier appui sur la flèche du haut affiche la ligne de commande **pwd**, un second la commande **date**. La flèche du bas navigue dans l'autre sens, jusqu'à l'invite d'origine. Si vous appuyez sur la touche [Entrée] vous lancez de nouveau la commande.

Les commandes

- **Rappel de l'historique**

Plus vous tapez des commandes, plus l'historique s'agrandit. Le shell conserve ainsi un grand nombre d'entrées dans l'historique (le nombre de lignes conservées peut être modifié). Cet historique est conservé dans un fichier caché de votre répertoire personnel appelé .bash_history. Vous pouvez voir le contenu de l'historique avec la commande **history.**

La commande **fc** effectue presque la même chose lorsqu'on utilise le paramètre -l. Par défaut elle se limite aux quinze dernières commandes. Aussi vous pouvez lui passer le nombre des dernières commandes, comme ceci pour les dix dernières : **fc -l -10 .**