Systèmes d'exploitation

« Il est plus facile de définir un système d'exploitation par ce qu'il fait que par ce qu'il est. »
J.L. Peterson.
« Un système d'exploitation est un ensemble de procédures cohérentes qui a pour but de gérer la pénurie de ressources. » J-l. Stehlé P. Hochard.

Operating System

« Unix est convivial. Cependant Unix ne précise pas vraiment avec qui. »
Steven King

« Unix ne dit jamais 's'il vous plaît'. »
Rob Pike

« Unix n'a pas été conçu pour empêcher ses utilisateurs de commettre des actes stupides, car cela les empêcherait aussi des actes ingénieux. »
Doug Gwyn

« Unix est simple. Il faut juste être un génie pour comprendre sa simplicité. »
Denis Ritchie

Objectifs

- Se familiariser avec les concepts fondamentaux utilisés dans les systèmes d'exploitation

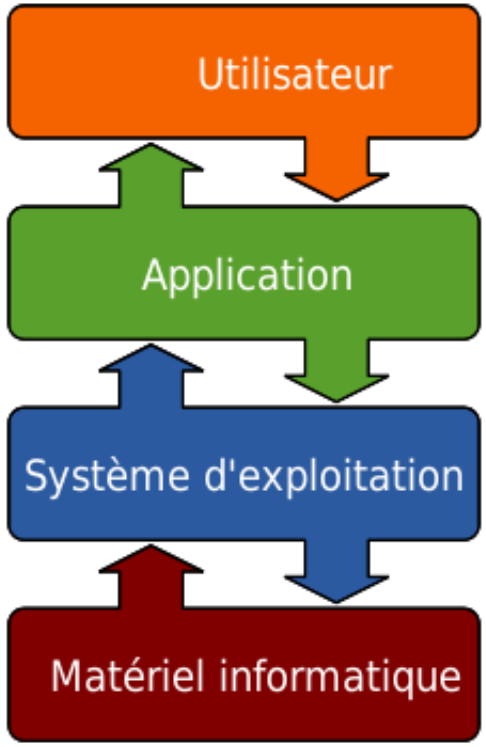
- Acquérir les capacités d'auto-formation nécessaires pour suivre les évolutions à venir

Définitions

- Le système d'exploitation (SE ou OS pour *operating system*) est l'ensemble de programmes central d'un équipement informatique qui sert d'interface entre le matériel et les logiciels applicatifs.

- C'est donc une couche logicielle (*software*) qui permet et coordonne l'utilisation du matériel (*hardware*) entre les différents programmes d'application.

- Un système d'exploitation est typiquement composé : d'un noyau, de bibliothèques, d'un ensemble d'outils système et de programmes applicatifs de base.



Naissance

- Dans les années 1940, les ordinateurs étaient programmés par manipulation d'interrupteurs à bascule, puis en introduisant une pile de cartes perforées dans un lecteur.
- Les ordinateurs des années 1940 à 1960, très coûteux, étaient la propriété des entreprises et des institutions. Chaque utilisateur avait le droit d'utiliser l'ordinateur pendant un temps limité et l'utilisateur avait alors à disposition la totalité du matériel de l'ordinateur. Il apportait avec lui une pile de cartes perforées qui contenait les instructions du ou des programmes à exécuter.
- Les ordinateurs de cette époque effectuaient une seule tâche à la fois, au service d'un seul utilisateur. Les programmes pour ces ordinateurs contenaient toutes les instructions nécessaires pour manipuler le matériel de l'ordinateur. Si la logithèque pour cet ordinateur comportait cinquante programmes, les instructions nécessaires pour manipuler le matériel se retrouvaient dans chacun de ces cinquante programmes.
- Avec l'agrandissement de la logithèque, l'idée est venue d'isoler les instructions routinières dans un programme séparé. Un programme qui résiderait continuellement dans la mémoire, quel que soit le programme en cours d'exécution. Ce programme était un système d'exploitation, sous sa forme la plus rudimentaire.

Historique (I)

- 1965 : le MIT (Massachusetts Institute of Technology) se lance dans la création du premier système d'exploitation multi-tâches et multi-utilisateurs : Multics (MULTiplexed Information and Computing Service ou service multiplexé d'information et de calcul).
- 1969 : les ingénieurs Ken Thompson et Dennis Ritchie des laboratoires Bell se lance dans l'écriture d'une version allégée de Multics. Le système, fonctionnel, est surnommé Unics, puis finalement baptisé UNIX. Rapidement reprogrammé dans un langage de programmation plus approprié (le C, développé par Ritchie pour l'occasion), UNIX se révèle particulièrement simple à porter sur de nouvelles plateformes, ce qui assure son succès.
- 1972 : le Micral de la société française R2E est le premier micro-ordinateur du monde. il est équipé d'un processeur Intel 8008 et du système d'exploitation SYSMIC (appelé alors moniteur d'exploitation). Le système d'exploitation SYSMIC sera plus tard renommé PROLOGUE lors du rachat de la société par Bull en 1978.
- 1980 : CP/M (Control Program/Monitor) est un système d'exploitation créé par Gary Kildall de Digital Research Inc. Il sera utilisé notamment sur les Amstrad CPC, Commodore 128, TRS-80, le ZX Spectrum. Les premières versions de MS-DOS se sont largement inspirées de CP/M.

Historique (II)

- 1980 : IBM prend contact avec Bill Gates, co-fondateur de la société Microsoft, pour l'adaptation du langage BASIC à son nouveau micro-ordinateur, le Personal Computer (PC). IBM est également à la recherche d'un système d'exploitation, et Gates conseille à la société de se tourner vers CP/M. Mais Gary Kildall refuse de signer le contrat avec IBM. Bill Gates saute sur l'occasion : il rachète QDOS (un système d'exploitation quick-and-dirty pour les processeurs Intel 8086) pour proposer à IBM le package DOS/BASIC. Après quelques modifications effectuées à la demande d'IBM, le système est baptisé MS-DOS.
- 1987 : Andrew Tanenbaum, professeur à l'université libre d'Amsterdam a créé le système d'exploitation Minix, clone d'UNIX dont le code source était destiné à illustrer son cours sur la construction des systèmes d'exploitation
- 1991 : Inspiré par les travaux de Tanenbaum, Linus Torvalds, étudiant à l'université d'Helsinki se lance alors dans le développement de son propre noyau : Linux, qui est à la base une réécriture de Minix. La toute première version (0.01) sort en 1991, Linux passe sous licence GNU en 1992 et il faut attendre 1994 pour voir la version 1.0 qui donne naissance à la distribution d'un système d'exploitation entièrement libre, GNU/Linux.

De nos jours ...

- Depuis 2010 les deux familles de systèmes d'exploitation les plus populaires sont Unix (dont Mac OS X et Linux) et Windows.
- La gamme des systèmes Windows équipe aujourd'hui 38 % des serveurs et 90 % des ordinateurs personnels, ce qui la place en situation de monopole notamment auprès du grand public. En 2008 ses parts de marché sont descendues en dessous de 90 % pour la première fois depuis 15 ans.
- La famille de systèmes d'exploitation Unix compte plus de 25 membres et les parts de marché de ces systèmes d'exploitation Unix sont de presque 50% sur les serveurs. La famille Unix anime 60 % des sites web dans le monde et Linux équipe 95 % des 500 super-ordinateurs du monde.
- On dénombre plus d'une centaine de systèmes d'exploitation dans le monde (voir http://fr.wikipedia.org/wiki/Liste_des_syst%C3%A8mes_d%27exploitation)

Remarques

- De nombreux logiciels applicatifs sur le marché sont construits pour fonctionner avec un système d'exploitation en particulier, ou une famille en particulier.
- Un système d'exploitation est construit pour fonctionner avec une gamme de machines donnée (type de processeur, constructeur, architecture).
- Pour l'acheteur le choix de la famille de machine limite le choix du système d'exploitation, qui lui-même limite le choix des logiciels applicatifs.
- L'utilité d'un système d'exploitation pour l'usager accroît avec le nombre de logiciels applicatifs qui sont prévus pour lui.
- La popularité élevée d'un système d'exploitation attire les éditeurs de logiciels applicatifs, ce qui accroît encore sa popularité. Ce phénomène fait que le marché est sujet aux situations de monopole.
- Les systèmes d'exploitation sont souvent vendus avec les appareils informatiques.
- Apple, Sun Microsystems et Silicon Graphics sont des marques qui fabriquent du matériel informatique et développent des systèmes d'exploitation pour leur propre matériel.

Typologie

- Un système d'exploitation est dit **multi-tâches** quand il permet l'exécution simultanée de plusieurs programmes. Tous les systèmes d'exploitation actuels sont multi-tâches.

- Il est dit **multi-utilisateurs** quand il est conçu pour être utilisé simultanément par plusieurs usagers, souvent à travers un réseau informatique (notion de serveurs). Ils sont multi-tâches et en général sécurisés, c'est-à-dire qu'il vont refuser d'exécuter toute opération pour laquelle l'usager n'a pas préalablement reçu une permission.

- Il est dit **multi-processeurs** quand il est conçu pour exploiter un ordinateur équipé de plusieurs processeurs. Dans de tels systèmes d'exploitation, plusieurs programmes sont exécutés simultanément par les différents processeurs.

- Il est dit **temps réel** quand il garantit que les opérations seront effectuées en respectant des délais stricts, et ce quelles que soient les conditions d'utilisation (charge du système). De tels systèmes d'exploitation sont utilisés dans l'industrie, l'aéronautique ou l'électronique pour créer des systèmes temps réel (souvent embarqué).

Besoins

- Un système d'exploitation est composé d'une large palette de programmes. Sa composition exacte dépend de l'usage cible et du type d'équipement informatique (ordinateur personnel, serveur, super-ordinateur ou encore système embarqué) auquel le système est destiné.

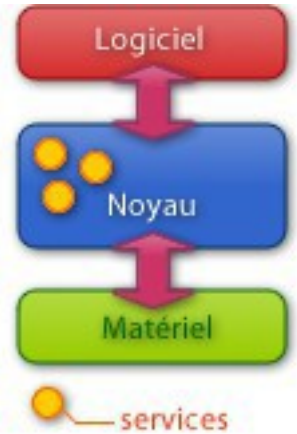
- Par exemple, si le système d'exploitation est destiné à être utilisé :
  - sur un ordinateur personnel ou une console de jeu vidéo, il nécessitera une interface graphique ergonomique
  - sur un serveur, il comprendra une large palette de protocoles et de pilotes pour du matériel réseau, sera multi-tâches et muni de contrôles d'accès
  - sur un assistant personnel ou un téléphone portable, le nombre de pilotes sera restreint au minimum et le système d'exploitation sera prévu pour être enregistré sur une mémoire morte
  - sur des super-ordinateurs, ils sera alors massivement multiprocesseur

- On pourra aussi distinguer l'usage qui en est fait :

  ☹ **Utilisateurs** ☺ **Programmeurs** ☺ **Administrateurs**

Fonctionnalités

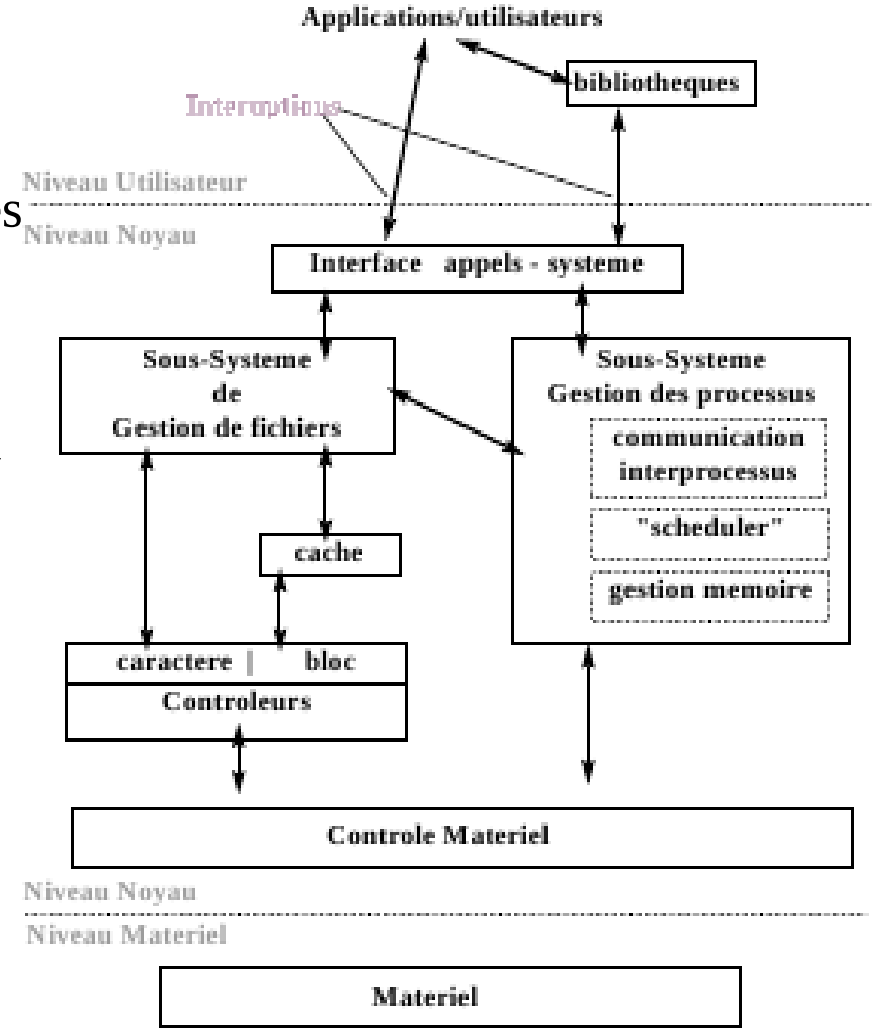
- Le système d'exploitation :
  - permet l'exploitation des périphériques matériels dont il coordonne et optimise l'utilisation
  - propose aux logiciels applicatifs des interfaces de programmation standardisées qui simplifient l'utilisation des matériels et des services qu'il offre
  - réalise enfin différentes fonctions visant à assurer la fiabilité (tolérance aux pannes, isolation des fautes) et la sécurité informatique (traçabilité, confidentialité, intégrité et disponibilité)



- Un système d'exploitation peut servir aussi :
  - à coordonner l'utilisation du ou des processeur(s), et accorder un certain temps pour l'exécution de chaque processus
  - à réserver de l'espace dans les mémoires pour les besoins des programmes
  - à organiser le contenu des disques durs ou d'autres mémoires de masse en fichiers et répertoires (dossiers)
  - à fournir les interfaces homme-machine des différents programmes
  - à recevoir les manipulations effectuées par l'utilisateur via le clavier, la souris ou d'autres périphériques, et les transmettre aux différents programmes.

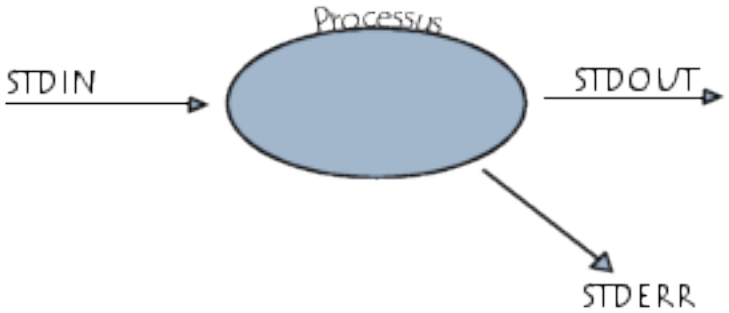
Architecture

- Un système d'exploitation est typiquement organisé en couches distinctes.
- La couche supérieure est l'interface de programmation avec les logiciels applicatifs (dont font partie les logiciels utilitaires et les bibliothèques fournis avec le système d'exploitation).
- Au centre, on trouve une ou plusieurs couches qui contiennent les composants principaux du système d'exploitation tels que : la gestion des systèmes de fichiers et du réseau, la gestion de mémoire, les pilotes, l'ordonnanceur, le gestionnaire d'interruption.
- La couche inférieure, appelée couche d'abstraction matérielle HAL (*Hardware Abstraction Layer*), est chargée de masquer les particularités matérielles.



Structure User/Kernel

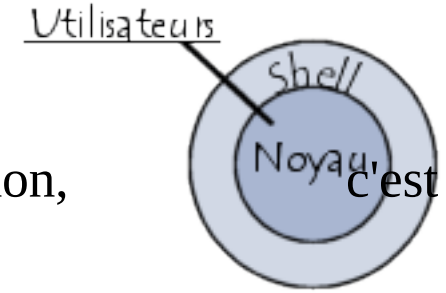
- *User* : Espace d'exécution des processus
  Un processus (*process*) est l'exécution d'un programme (binaire exécutable) dans un environnement, pour le compte d'un utilisateur.



- *Kernel* : Espace d'exécution de l'OS
  Le noyau (*kernel*) est un espace mémoire isolé et protégé, dans lequel est placé tout ou partie du système d'exploitation. Dans le langage courant le terme *kernel* désigne l'emplacement ainsi que l'ensemble des programmes qu'il contient et qui forment le cœur rigide du système d'exploitation. On distinguera plus tard : noyau monolithique, micro-noyau, noyau hybride, exo-noyau et noyau temps réel.

Interface utilisateur

- L'interface homme-machine (IHM) permet à un utilisateur de dialoguer avec la machine.



- Le *shell* (coquille) est la partie la plus externe du système d'exploitation, c'est l'interface utilisateur du système d'exploitation.
- On distingue deux types d'IHM ou de *shell* :
  - GUI (*Graphical User Interface*) ou « interface utilisateur graphique » : les parties les plus typiques de ce type d'environnement sont le pointeur de souris, les fenêtres, le bureau, les icônes, les boutons, les menus, les barres de défilement, ... Les systèmes d'exploitation grand public (Windows, MacOS, GNU/Linux, etc.) sont pourvus d'une interface graphique qui, dans un soucis d'ergonomie, se veut conviviale, simple d'utilisation et accessible au plus grand nombre pour l'usage d'un ordinateur personnel.
  - CLI (*Command Line Interface*) ou « interface en ligne de commande » est encore utilisée en raison de sa puissance, de sa grande rapidité, son uniformité, sa stabilité et du peu de ressources nécessaires à son fonctionnement. Le système d'exploitation permet cette possibilité par l'intermédiaire d'un interpréteur de commandes (le *shell*). Beaucoup de serveurs ne s'administrent qu'en ligne de commande.

Environnement fenêtré

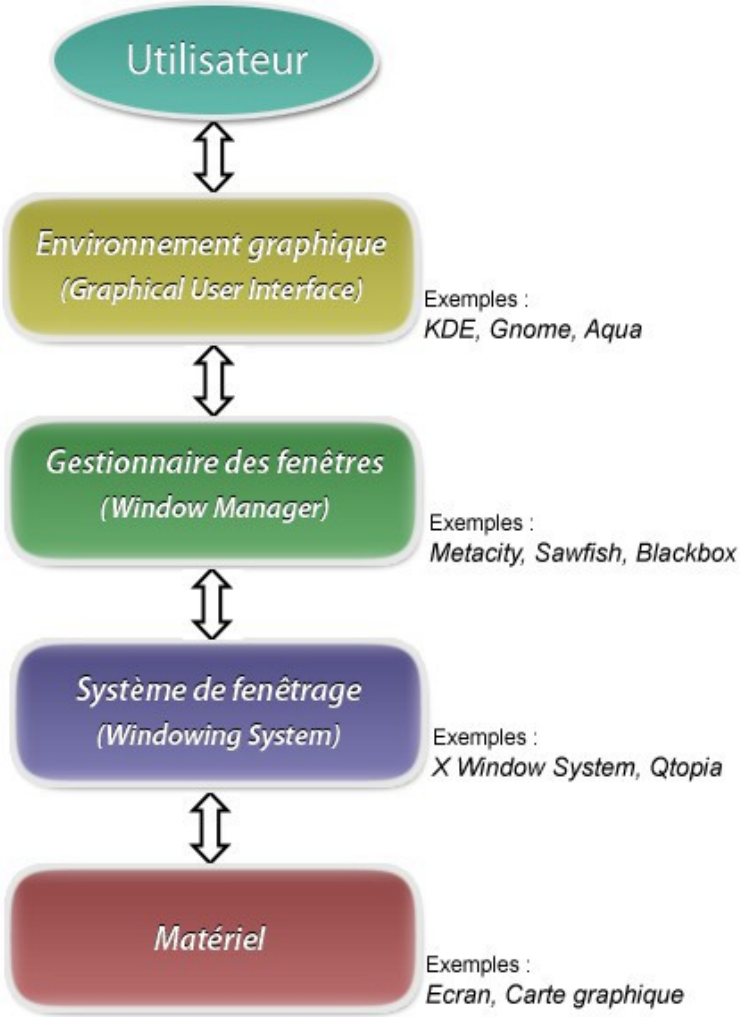
- Historique :
  - 1968 : Douglas Engelbart montre pour la première fois un environnement graphique comportant des fenêtres à manipuler avec une souris.
  - 1973 : Le Xerox Alto, ordinateur conçu au Xerox PARC, fut un des premiers ordinateurs et le premier à utiliser la métaphore du bureau et une interface graphique.
  - 1981 : Environnement graphique X Window développé au MIT (*Massachusetts Institute for Technology*)

- Aussi appelé WIMP (*Windows* (fenêtres), *Icons* (icônes), *Menus* (menus) *and Pointing device* (dispositif de pointage)), ce type d'interface graphique a été inventé par la firme Xerox et rendu célèbre par le Macintosh.
- Les parties les plus typiques d'un environnement fenêtré sont le concept de bureau pour représenter l'espace de travail, de fenêtre pour contrôler les programmes et manipuler les contenants (répertoires appelés dossiers), d'icônes pour symboliser les fichiers (appelés documents).
- Plusieurs contrôles graphiques sont couramment utilisés pour interagir avec l'utilisateur, comme les boutons, les menus, les listes déroulantes, les ascenseurs, etc.

Environnement graphique GNU/Linux

- Sous GNU/Linux et plus généralement tous les systèmes Unix, l'environnement graphique est traditionnellement séparé en plusieurs composants :

  1. Un serveur graphique (généralement *X Window System*) chargé des primitives de dessin de bas niveau ;
  2. Un gestionnaire de fenêtres comme Metacity, Enlightenment, KWin, Window Maker, Compiz Fusion (3D), … chargé de partager le serveur graphique entre les différentes applications ;
  3. Une bibliothèque de composants comme Qt, GTK+, wxWidgets ou Motif chargée de la gestion des différents composants de l'interface : bouton, ascenseurs, zone de texte, …



- Aujourd'hui les environnements de bureau regroupent un environnement graphique et un gestionnaire de fenêtres : GNOME, KDE, Xfce, GNUstep, CDE, ...

Gestionnaire de bureau

- Un gestionnaire permet d'offrir plusieurs fonctionnalités :

  - des barres des tâches ou un *dock* d'applications.
  - des menus, voir des menus fixables (*pinnable* menus en anglais).
  - des bureaux virtuels et donc des *workspace manager* (application qui permet de passer d'un bureau à l'autre; il est parfois appelé *pager*).
  - des icônes et donc le support d'un format d'icône (comme le XPM).
  - le déplacement, le redimensionnement, l'iconification des fenêtres.
  - la modification du fond d'écran, des couleurs des tous les éléments visibles, de l'habillage des fenêtres.
  - un *session management* qui permet de garder une trace des actions effectuées par un utilisateur.

- Les interfaces graphiques d'OS connus :
  - Mac OS X utilise Aqua
  - Atari ST utilise le GEM
  - Windows Vista et Windows 7 utilisent Aero

Interpréteur de commande Windows

- cmd.exe est l'interpréteur de commande en mode texte de Windows. Il est l'équivalent du *shell* sous Unix. Il peut interpréter des fichiers *batch* qui sont des fichiers textes contenant une série de commandes. Les fichiers *batch* de Windows sont des fichiers dotés de l'extension .BAT (ou .CMD).
- Sous Windows, l'appellation "*shell*" regroupe deux concepts :
  - Comme pour Unix, l'interpréteur en ligne de commande (command.com pour les versions de Windows basées sur MS-DOS, et cmd.exe pour celles qui reposent sur Windows NT)
  - L'interface graphique, en général l'*Explorer*.
- *Remarque :* avec Windows Vista est apparu un nouvel interpréteur de commande, le Windows *PowerShell* (utilisable aussi sous Windows XP), orienté objet.
- Lien : http://windows.developpez.com/cours/ligne-commande/
- En programmation Windows, on appelle aussi fonctions *shell* (ou *shell API*) les fonctions exportées par l'Explorer. Par exemple la fonction SHFileOperation() s'occupe des manipulations de fichiers (copie, déplacement, effacement…).

Interpréteur de commande UNIX/Linux

- Souvent nommé le *shell*, c'est un utilitaire qui sert d'interface entre la saisie des commandes et le noyau du SE. Il fait office d'IHM (Interface Homme-Machine).

- On parle aussi d'interpréteur de commandes du fait de sa fonction de base (lit et interprète les commandes).

- Il existe de nombreux *shell* sous UNIX/Linux. Dans la plupart des distributions, le *shell* par défaut est *bash* (*Bourne Again Shell*).

- Le shell permet :
  - l'exécution de commandes
  - la redirection des entrées et des sorties
  - la gestion des variables d'environnement
  - la possibilité de réaliser des scripts (programmation) pour l'automatisation de tâches (par exemple la sauvegarde)

Shell UNIX

- Sous UNIX/Linux, la ligne de commande a toujours été le moyen privilégié de communication avec l'ordinateur. Le Bourne shell (sh) est l'interpréteur originel de l'environnement UNIX.

- Celui-ci est utilisable en conjonction avec un terminal (souvent virtuel). A l'origine, l'interpréteur de commandes par défaut était sh, qui évolua en de nombreuses versions, dont csh, étendu en tcsh, ou ksh, ou encore rc... Mais aujourd'hui bash est le shell le plus répandu, bien qu'il existe d'autres interpréteurs de commandes, comme zsh, ou ash.

- L'invite (*prompt*) consiste en quelques caractères en début de ligne (généralement, le nom de compte de l'utilisateur, et/ou l'unité logique par défaut, et/ou le chemin par défaut, et/ou date, ...) se terminant par un caractère bien connu (souvent « $ », « # » ou « > »), invitant l'utilisateur à taper une commande.

- D'autres langages de scripts tels que Perl, Python ou Ruby, remplacent progressivement les interpréteurs qui sont encore prédominants dans les environnements de démarrage de systèmes UNIX.

Philosophie UNIX

- Résumer la philosophie d'UNIX n'est pas chose évidente. Il s'agit d'un ensemble de principes. Nombreux sont ceux qui ont essayé de les résumer ou les lister (taper « philisophie UNIX » ou « less is more » dans un moteur de recherche).

  - Des programmes qui effectuent une seule chose et qui le font bien
  - Le silence est d'or
  - Des programmes qui collaborent
  - Des programmes pour gérer des flux texte

« L'univers a 40 ans »

Systèmes de fichiers

- Un système de fichiers (*file system*) définit l'organisation d'un disque (ou partition d'un disque).
- C'est une structure de données permettant de stocker les informations et de les organiser dans des fichiers sur ce que l'on appelle des mémoires secondaires (disque dur, disquette, CD-ROM, clé USB, disques SSD, etc.).
- Un système de fichiers offre à l'utilisateur une vue abstraite sur ses données (fichiers) et permet de les localiser à partir d'un chemin d'accès.
- Le fichier est la plus petite entité logique de stockage sur un disque.
- Le formatage (action de formater) prépare un support de données de stockage en y inscrivant un système de fichiers, de façon à ce qu'il soit reconnu par le système d'exploitation de l'ordinateur.
- Une partition est une partie d'un disque dur destinée à accueillir un système de fichiers.
- Il existe de nombreux systèmes de fichiers différents : FAT, NTFS, HFS, ext2, ext3, UFS, etc.

Partitionnement

- Le partitionnement est un fractionnement d'un disque dur réel (matériel) en plusieurs disques virtuels (logiciels).
- Chaque partition possède donc son système de fichiers, qui permettra de stocker ensuite les données.
- On nomme « partition d'amorçage », celle qui prend le contrôle au démarrage, qu'elle contienne ou non le système d'exploitation.
- Les informations sur les partitions sont conservées sur le disque lui-même dans des tables de partitions. La table de partitions principale est contenue dans le premier secteur du disque ou secteur d'amorçage MBR (*Master Boot Record*) qui contient également le programme d'amorçage.
- Chaque ligne d'une table de partitions contient l'adresse de début de la partition et sa taille. Il peut s'agir de partitions primaires qui contiendront un système de fichiers ou de partitions étendues qui contiendront à leur tour une table de partitions ayant la même structure que la table principale.
- Un chargeur d'amorçage (ou *bootloader*) est un logiciel permettant de lancer (charger et démarrer) un système d'exploitation éventuellement parmi plusieurs (*multi-boot*).

Système de gestion des fichiers

- Le système de gestion des fichiers (SGF) assure plusieurs fonctions :
  - Manipulation des fichiers : des opérations sont définies pour permettre la manipulation des fichiers par les programmes d'application.
  - Allocation de la place sur mémoires secondaires : les fichiers étant de taille différente et cette taille pouvant être dynamique, le SGF alloue à chaque fichier un nombre variable de blocs de taille fixe.
  - Localisation des fichiers : il est nécessaire de pouvoir identifier et retrouver les données ; pour cela, chaque fichier possède un ensemble d'informations descriptives regroupées dans un descripteur de fichier.
  - Sécurité et contrôle des fichiers : le SGF permet le partage des fichiers par différents programmes d'applications tout en assurant la sécurité et la confidentialité des données. Le SGF se doit aussi de garantir la conservation des fichiers en cas de panne du matériel ou du logiciel.
- Remarque : il existe d'autres façons d'organiser les données, par exemple les bases de données et les fichiers indexés.

Fonctionnalités supplémentaires des FS

- Les systèmes de fichiers peuvent inclure :
  - la compression
  - le chiffrement automatique des données
  - une gestion plus ou moins fine des droits d'accès aux fichiers
  - une journalisation des écritures (robustesse en cas de défaillance du système)
  - une distribution ou répartition réseau
  - une virtualisation
  - des fonctionnalités pour le temps réel
  - des restrictions de nommage pour les fichiers

Notion de fichier

- Un fichier est une suite d'octets portant un nom et conservé dans une mémoire.
- Le contenu du fichier peut représenter n'importe quelle donnée binaire : un programme, une image, un texte, etc.
- Les fichiers sont classés dans des groupes appelés répertoires, chaque répertoire peut contenir d'autres répertoires, formant ainsi une organisation arborescente appelée système de fichiers.
- Les fichiers sont la plupart du temps conservés (stockés) sur des mémoires de masse tels que les disques durs.
- Dans un système d'exploitation multi-utilisateurs, les programmes qui manipulent le système de fichier effectuent des contrôles d'accès (notion de droits).

Caractéristiques des fichiers

- Quelques caractéristiques de base des fichiers :

- Le nommage et ses restrictions (nombre de caractères, caractères autorisés)
- Le chemin d'accès est une "formule" qui sert à indiquer l'emplacement où se trouve un fichier dans l'arborescence du système de fichier. La syntaxe diffère d'un système d'exploitation à l'autre.
- La taille du fichier indique la quantité d'informations conservée (exprimée en octets) en sachant que la taille physique (réellement occupée) est légèrement supérieure à la taille du fichier en raison de l'utilisation de blocs d'allocation de taille fixe.
- L'extension est un suffixe ajouté au nom du fichier pour indiquer la nature de son contenu. L'usage des extensions est une pratique généralisée sur les systèmes d'exploitation Windows et une pratique courante sur les systèmes d'exploitation Unix.
- Les données descriptives : la date de création et de modification, le propriétaire du fichier ainsi que les droits d'accès

Types de fichiers

- On distingue en général deux types de fichiers : texte et binaire
- Les fichiers texte ont un contenu pouvant être interprété comme du texte (une suite de bits représentant un caractère), la plupart du temps en codage ASCII. On utilise généralement un éditeur de texte pour manipuler ce type de fichiers. Quelques exemples de fichiers textes : code source d'un programme, fichiers de configuration, etc .
- Le contenu d'un fichier binaire correspond souvent à un format précis lié à l'usage d'un logiciel applicatif spécifique. Voici quelques exemples de formats binaires usuels : code machine (exécutable), fichier de base de données, fichiers multimédias : images, sons, vidéos, traitement de texte, etc.
- Un fichier (texte ou binaire) qui a subi une transformation par un algorithme en vue de diminuer sa taille est appelé fichier compressé. Le fichier transformé est un fichier binaire.
- Une archive est un fichier dans lequel se trouve regroupé des fichiers/répertoires ou tout le contenu d'une arborescence. Le but principal d'une archive est de tout contenir (données + descriptions) en un seul fichier. Les archives sont souvent compressés.

Le fichier dans son FS

- Chaque fichier est vu par le système de fichiers de plusieurs façons :
  - un descripteur de fichier (souvent un entier unique) permettant de l'identifier
  - une entrée dans un répertoire permettant de le situer et de le nommer
  - des métadonnées sur le fichier permettant de le définir et de le décrire
  - un ou plusieurs blocs (selon sa taille) permettant d'accéder aux données du fichier (son contenu)
- Métadonnées : des données servant à définir ou décrire d'autres données

Inode

- Le terme inode désigne le descripteur d'un fichier sous UNIX/Linux. Les inodes (contraction de « index » et « node », en français : nœud d'index) sont des structures de données contenant des informations concernant les fichiers stockés dans certains systèmes de fichiers (notamment de type Linux/Unix). À chaque fichier correspond un numéro d'inode (*i-number*) dans le système de fichiers dans lequel il réside, unique au périphérique sur lequel il est situé.
- Les métadonnées les plus courantes sous UNIX sont :
  - droits d'accès en lecture, écriture et exécution selon l'utilisateur, le groupe, ou les autres ;
  - dates de dernier accès, de modification des métadonnées (*inode*), de modification des données (*block*) ;
  - propriétaire et groupe propriétaire du fichier ;
  - taille du fichier ;
  - nombre d'autres inodes (liens) pointant vers le fichier ;
  - nombre et numéros de blocs utilisés par le fichier;
  - type de fichier : fichier simple, lien symbolique, répertoire, périphérique, etc.
- Sur la plupart des systèmes Unix, la commande *stat* permet d'afficher l'intégralité du contenu de l'inode.

Arborescence

- Pour l'utilisateur, un système de fichiers est vu comme une arborescence : les fichiers sont regroupés dans des répertoires (concept utilisé par la plupart des systèmes d'exploitation).

- Ces répertoires contiennent soit des fichiers, soit d'autres répertoires. Il y a donc un répertoire racine et des sous-répertoires.

- Une telle organisation génère une hiérarchie de répertoires et de fichiers organisés en arbre.

- Sous Unix/Linux, les utilisateurs voient une arborescence de fichiers unique (/). Cet arbre est en fait l'unification de plusieurs systèmes de fichiers.

- Dans un système Windows, les périphériques de stockage de données et les partitions sont affichés comme des lecteurs indépendants (C:, D:, ...) en haut de leur propre arborescence (\).

Chemin d'accès

- Le chemin d'accès (*PATH*) permet d'atteindre un fichier dans son système de fichiers (FS).
- On distingue :
  - Chemin absolu : à partir de la racine du FS
  - Chemin relatif : à partir du répertoire courant
- Il y a deux répertoires particuliers : le répertoire courant (.) et le répertoire parent (..)
- Exemple sous Windows :
  C:\WINNT\system\sys.ini : désignation absolue du fichier sys.ini se trouvant dans le répertoire C:\WINNT\system\ ;
  sys.ini : désignation relative du fichier sys.ini, lorsque l'utilisateur se trouve dans le répertoire C:\WINNT\system\.
- Exemple sous Unix/Linux :
  /var/spool/mail/r4f : désignation absolue du fichier r4f se trouvant dans le répertoire /var/spool/mail ;
  ./mail/r4f : désignation relative du fichier r4f lorsque l'utilisateur se trouve dans le répertoire /var/spool/.

Contrôle d'accès

- Dans un système d'exploitation multi-utilisateurs, tout utilisateur doit préalablement décliner son identité avant d'utiliser l'ordinateur (procédure d'authentification ou de *login*).
- Puis un programme du système d'exploitation vérifie cette identité par rapport à un annuaire ou un référentiel. Même si les utilisateurs possèdent un nom (*login*), le système manipule des identifiants numériques UID (*User IDentifier*).
- Lors de chaque opération demandée par un logiciel applicatif, le système d'exploitation vérifie préalablement si l'utilisateur qui exécute ce processus est autorisé à effectuer cette opération. La vérification se fait sur la base des règlements (*policies*) ainsi que des listes de droits d'accès (ACL) introduits par l'administrateur.
- Le système d'exploitation refusera toute opération non autorisée et inscrira le refus dans un journal d'activité (*log*).
- Les mécanismes de contrôle d'accès ont aussi pour effet de lutter contre les logiciels malveillants. Un logiciel malveillant (*malware*) est un logiciel développé dans le but de nuire à un système informatique. Les virus et les vers sont deux exemples de logiciels malveillants les plus connus, en revanche il en existe beaucoup d'autres.

Notions de session

- Une session est une période délimitée pendant laquelle un équipement informatique est en communication et réalise des opérations au service d'un client (un utilisateur), un logiciel ou un autre équipement.

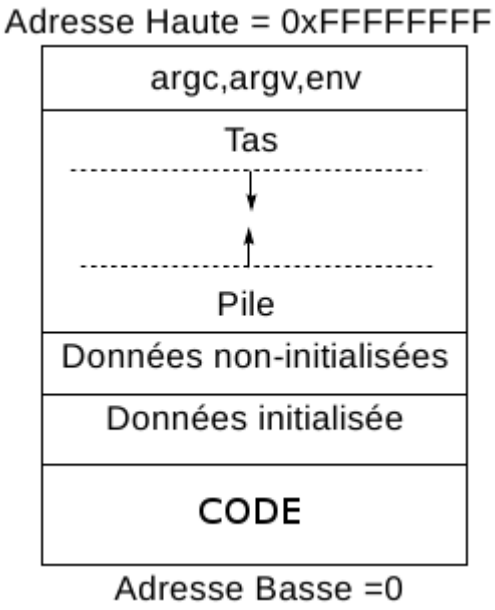
- Dans les systèmes d'exploitation multi-utilisateurs, elles permettent à chacun de travailler indépendamment des autres dans des processus distincts, avec ses propres variables d'environnement.

- Les informations de session sont conservées dans un profil de l'utilisateur (en général, un ensemble de variables qui peuvent être stockées dans des fichiers).

- Une session démarre lorsque l'utilisateur commence à interagir avec un programme. L'ouverture de session peut être explicite, avec une saisie d'un nom (*login*) et d'un mot de passe. On parle alors d'authentification ou de connexion.

Processus

- Rappel : Un programme qui s'exécute est appelé un processus.
- Un processus comporte du code machine exécutable, une zone mémoire (données allouées par le processus), une pile ou *stack* (pour les variables locales des fonctions et la gestion des appels et retour des fonctions) et un tas ou *heap* pour les allocations dynamiques.
- Ce processus est une entité qui, de sa création à sa mort, est identifié par une valeur numérique : le PID (*Process IDentifier*).
- Tous les processus sont donc associés à une entrée dans la table des processus qui est interne au noyau.

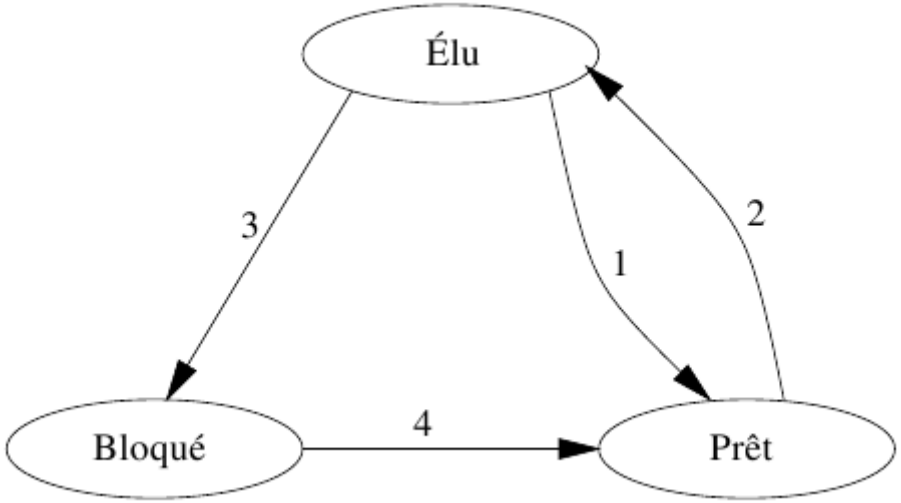


- Chaque processus a un utilisateur propriétaire, qui est utilisé par le système pour déterminer ses permissions d'accès aux fichiers.
- Remarques : Les commandes ps et top listent les processus sous UNIX/Linux et, sous Windows on utilisera le gestionnaire de tâches (taskmgr.exe).

États d'un processus

- Le processus est une activité dynamique et possède un état qui évolue au cours du temps.

- Ce processus transitera par différents états selon que :
  - il s'exécute (ACTIF ou Élu)
  - il attend que le noyau lui alloue le processeur (PRET)
  - il attend qu'un événement se produise (ATTENTE ou Bloqué)

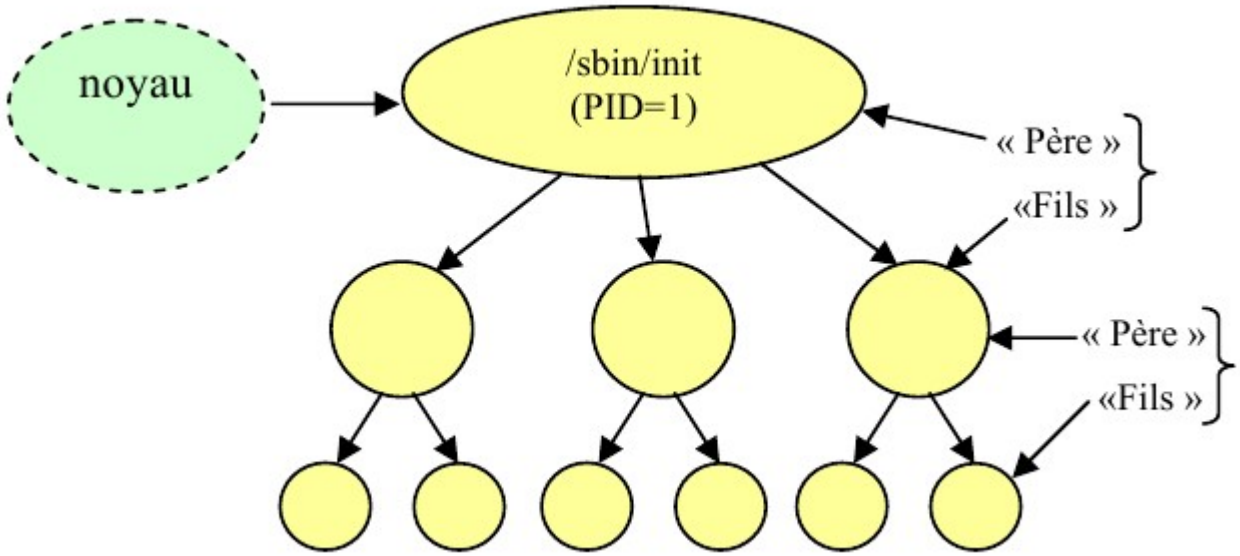


- C'est l'ordonnanceur (*scheduler*) qui contrôle l'exécution et donc les états des processus.

Généalogie des processus

- La création d'un processus étant réalisée par un appel système (*fork* sous UNIX/Linux), un processus est forcément créé par un autre processus (notion de père-fils).

- Exemple sous UNIX/Linux :



Bibliothèque

- Une bibliothèque ou librairie logicielle est un ensemble de fonctions utilitaires, regroupées et mises à disposition afin de pouvoir être utilisées sans avoir à les réécrire.

- Les bibliothèques logicielles se distinguent des exécutables dans la mesure où elles ne sont pas complètes car elles ne possèdent pas de fonction principale (*main*) et par conséquent elles ne peuvent pas être exécutées directement.

- Les bibliothèques logicielles sont parfois regroupées en un *framework*, de façon à constituer un ensemble cohérent et complémentaire de bibliothèques.

- L'interface de programmation ou API (*Application Programming Interface*), est la partie visible d'une bibliothèque ou d'un ensemble de bibliothèques.

Bibliothèque statique et dynamique

- Une bibliothèque statique contient des fonctions qui seront intégrées au code généré par le compilateur (édition de liens statique). L'inconvénient est qu'un programme ainsi obtenu n'est pas mis à jour lorsque la bibliothèque change. L'avantage est que le programme seul est autonome.
- Une bibliothèque dynamique est une bibliothèque qui contient des fonctions qui seront intégrées au code au moment de son exécution (édition de liens dynamique). L'avantage est que le programme (plus léger) est à jour vis-à-vis de la mise à jour des bibliothèques. L'inconvénient est que l'exécution dépend de l'existence de la bibliothèque sur le système cible.
- Une bibliothèque dynamique, *Dynamic Link Library* (.dll) pour Windows et *shared object* (.so) sous UNIX/Linux, est un fichier de bibliothèque logicielle utilisé par un programme exécutable, mais n'en faisant pas partie. Ce fichier contient des fonctions qui pourront être appelées pendant l'exécution d'un programme, sans que celles-ci soient incluses dans son exécutable.

Interface de programmation

- *« La véritable "idée" d'un système d'exploitation est d'utiliser les fonctionnalités du matériel, et de les placer derrière une couche d'appels de haut niveau. » Linus Benedict Torvalds*
- Le noyau est donc vu comme un ensemble de fonctions (API) : chaque fonction ouvre l'accès à un service offert par le noyau. Ces fonctions sont regroupées au sein de la bibliothèque des appels systèmes (*system calls*) pour UNIX/Linux. On peut soit les utiliser par programmation (en C par exemple), soit en exécutant des commandes dans un *shell* ... soit en utilisant des applications (généralement à partir d'une interface graphique).
- L'utilisation de la même interface de programmation quel que soit le matériel, le protocole ou le système de fichier concerné assure la portabilité des logiciels applicatifs : un logiciel applicatif donné pourra fonctionner sur différents ordinateurs, quelle que soit leur configuration, en particulier quel que soit le matériel, le système de fichier ou le protocole utilisé.
- POSIX (*Portable Operating System Interface*) est une norme relative à l'interface de programmation du système d'exploitation. De nombreux systèmes d'exploitation sont conformes à cette norme, notamment les membres de la famille Unix.

Pilotes

- Un pilote (*driver*) est un programme qui contient les instructions à exécuter pour utiliser un certain périphérique informatique. Les pilotes sont fournis par le système d'exploitation ou le fabricant du périphérique. En général, chaque périphérique a son propre pilote. Sans pilote, l'imprimante ou la carte graphique par exemple ne pourraient pas être utilisées.

- L’interface de programmation est similaire sur tous les pilotes ce qui assure une utilisation transparente et banalisée du périphérique.

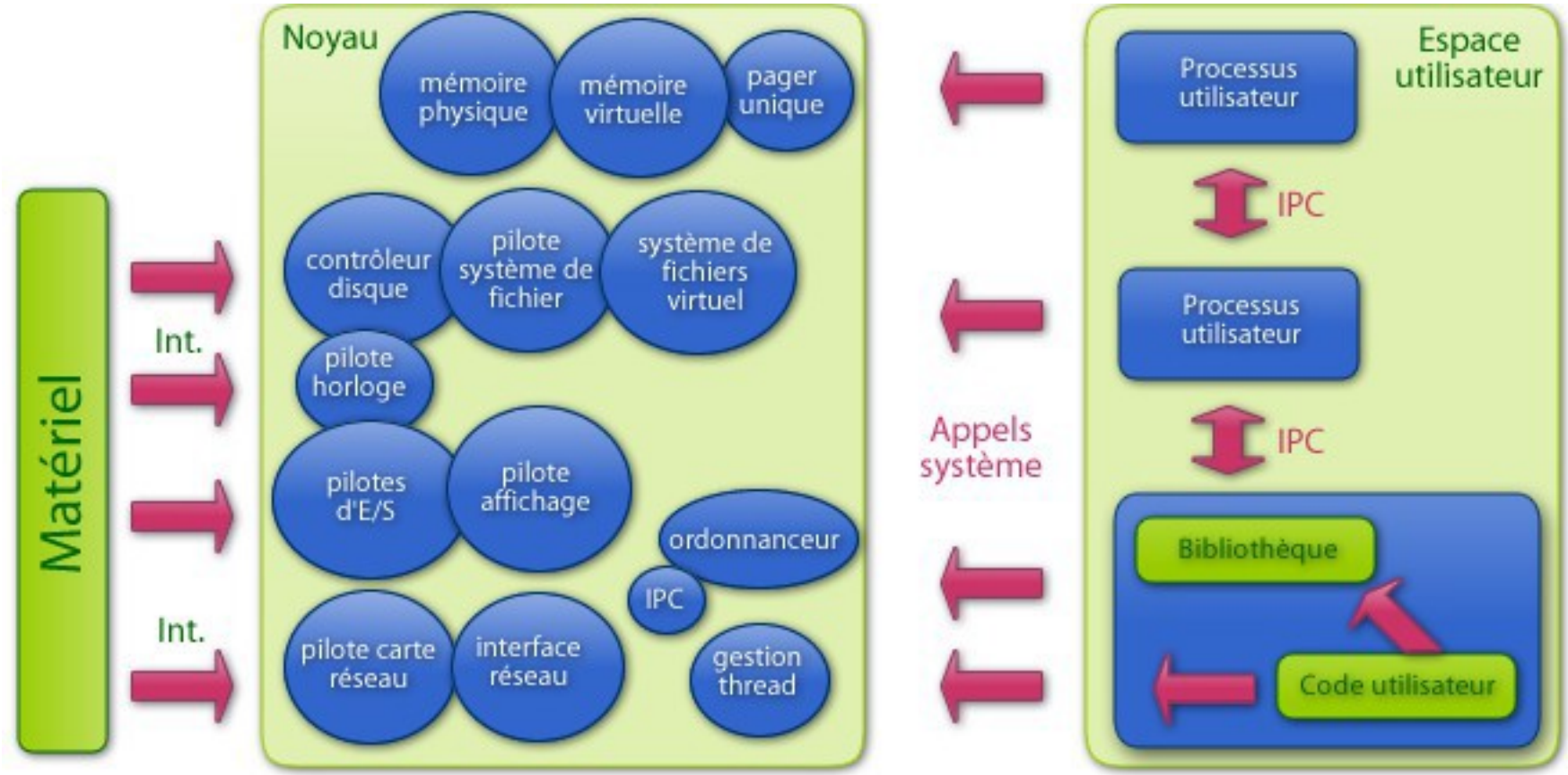
- Lorsqu'un événement (d'E/S) survient sur un périphérique (réception d'informations, fin d'un envoi, manipulation effectuée par l'utilisateur...), celui-ci génère une interruption matérielle. Le système d'exploitation déclenche alors l'exécution des instructions du pilote concerné (routine d'interruption).

- Les logiciels applicatifs ainsi que le système d'exploitation peuvent également, au besoin, déclencher l'exécution des instructions d'un pilote.

Différents types de noyaux

- Il existe toutes sortes de noyaux, plus ou moins spécialisés. L'ensemble de ces noyaux peut être divisé en deux approches opposées d'architectures logicielles : les noyaux monolithiques et les micro-noyaux.
- On considère généralement les noyaux monolithiques, de conception ancienne, comme obsolètes car difficiles à maintenir et moins « propres ». Le noyau Linux était déjà qualifié d'obsolète par Andrew Tanenbaum, dès sa création en 1991.
- La mise en place de micro-noyaux, qui consiste à déplacer l'essentiel des fonctions du noyau vers l'espace utilisateur, est très intéressante en théorie mais s'avère difficile en pratique. Ainsi les performances du noyau Linux (monolithique) sont supérieures à celles de ses concurrents (noyaux généralistes à micro-noyaux), sans compter qu'il est modulaire depuis 1995.
- Pour ces raisons de performance, les systèmes généralistes basés sur une technologie à micro-noyau, tels que Windows et Mac OS X, n'ont pas un « vrai » micro-noyau enrichi. Ils utilisent un micro-noyau hybride : certaines fonctionnalités qui devraient exister sous forme de mini-serveurs se retrouvent intégrées dans leur micro-noyau, utilisant le même espace d'adressage.
- Ainsi, les deux approches d'architectures de noyaux, les micro-noyaux et les noyaux monolithiques, considérées comme diamétralement différentes en termes de conception, se rejoignent quasiment en pratique par les micro-noyaux hybrides et les noyaux monolithiques modulaires.

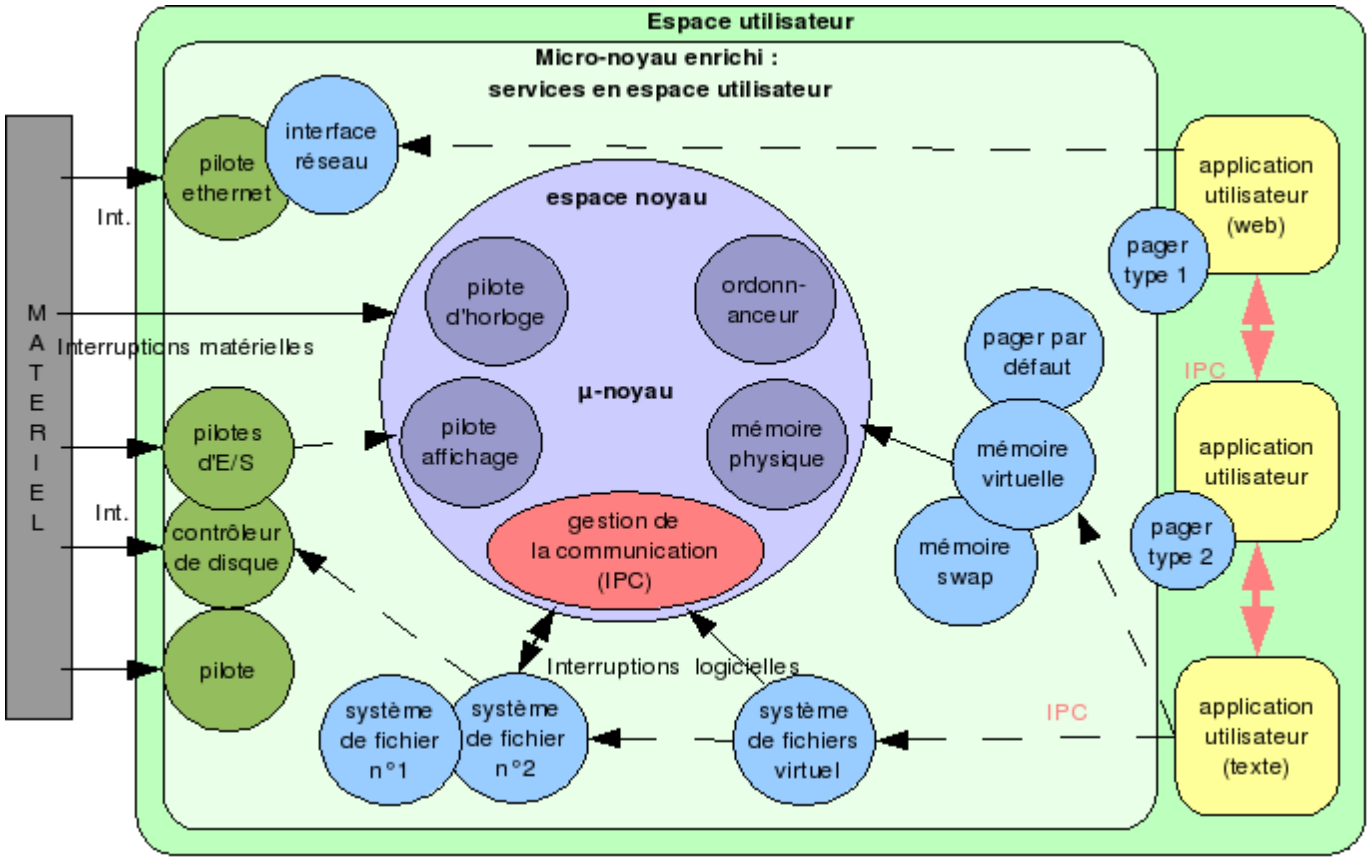
Noyaux monolithiques



- Pour répondre aux problèmes des noyaux monolithiques, ces derniers sont devenus modulaires : seules les parties fondamentales du système sont regroupées dans un bloc de code unique (monolithique). Les autres fonctions, comme les pilotes matériels, sont regroupées en différents modules chargeables directement dans le noyau à la demande de fonctionnalités et augmente les possibilités de configuration.

Micro-noyaux

Les systèmes à micro-noyaux cherchent à minimiser les fonctionnalités dépendantes du noyau en plaçant la plus grande partie des services du système d'exploitation à l'extérieur de ce noyau, c'est-à-dire dans l'espace utilisateur. Ces fonctionnalités sont alors fournies par de petits serveurs indépendants possédant souvent leur propre espace d'adressage.



En éloignant les services « à risque » à l'extérieur du noyau, il permet de gagner en robustesse et en fiabilité, tout en facilitant la maintenance et l'évolutivité. En revanche, les mécanismes de communication (IPC), qui deviennent fondamentaux pour assurer le passage de messages entre les serveurs, sont très lourds et peuvent limiter les performances.

Système embarqué

- Un système embarqué (ou système enfoui) peut être défini comme un système électronique et informatique autonome, qui est dédié à une tâche bien précise.

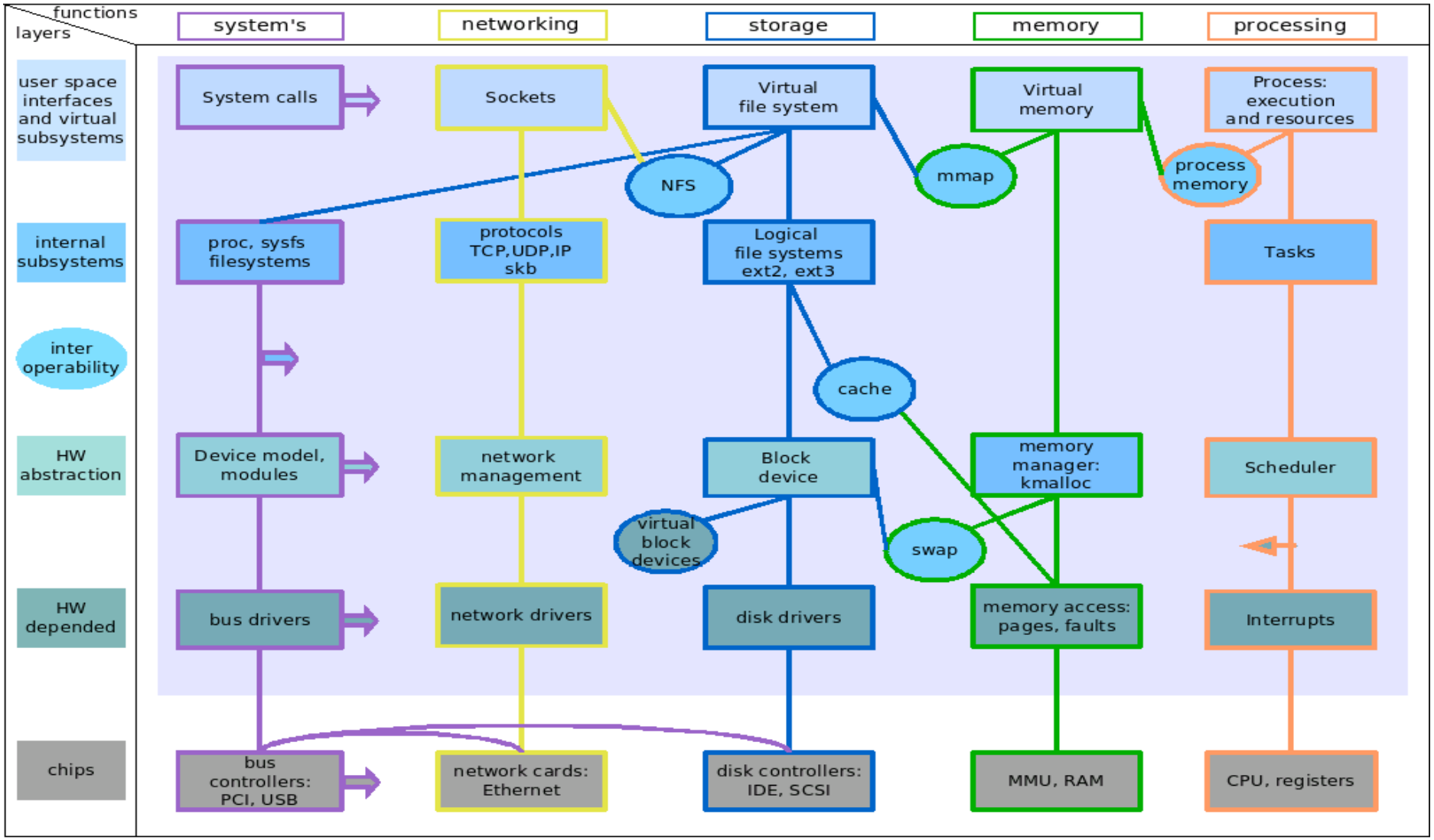
- Ses ressources disponibles sont généralement limitées (coût faible, taille réduite, consommation restreinte, espace mémoire limité de l'ordre de quelques Mo maximum, puissance de calcul juste nécessaire pour répondre aux besoins, ...).

- Il peut posséder un système d'exploitation (souvent temps réel).

- Il existe de nombreux OS spécifiques aux systèmes embarqués : Android, Cisco IOS, iOS (ou iPhone OS), Palm OS, Symbian OS, Windows CE, Windows Mobile, LynxOS, QNX, eCos, µCLinux, VxWorks, OS9, etc.

Exemple : le noyau Linux

Simplified Linux kernel diagram in form of a matrix map



Exemple : l'architecture Windows NT