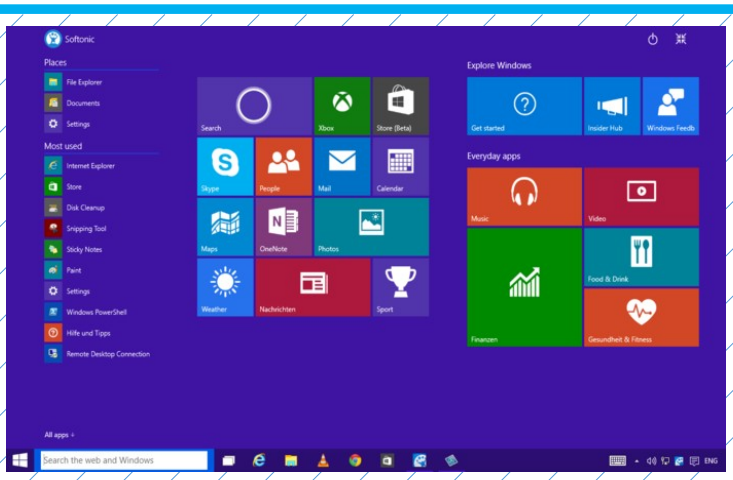
UNIVERSITE VIRTUELLE DE CÔTE D'IVOIRE



FILIERES

- Développement d'Applications et e-Services (DAS)
- Réseaux et Sécurité Informatique (RSI)
- Multimédia et Arts Numériques (MMX)
- Bases de Données (BD)

TS STIC /INP-HB

AUTEUR

Irié Bi Dizan Paul, Mamadou BAKOUAN

DATE : 19/08/2017

« Cette partie est la suite du programme d'informatique de la première année des prépas bio »

TABLE DES MATIERES

LECON 1 : GENERALITES SUR LES SYSTEMES D'EXPLOITATION

1.1 Objectifs

À la fin de cette leçon, vous serez capable de :

- ✓ Connaître la définition d'un système d'exploitation
- ✓ Connaître l'évolution des systèmes d'exploitation
- ✓ Connaître le rôle d'un système d'exploitation
- ✓ Connaître les classes des systèmes d'exploitation
- ✓ Connaître les différentes structures de système d'exploitation

1.2 Introduction aux systèmes d'exploitation (SE)

❖ **Définition d'un SE**

Les ordinateurs permettent de collecter des données, de réaliser des calculs, de stocker des informations et de communiquer avec d'autres ordinateurs. Un ordinateur est formé d'une partie matérielle et d'une partie logicielle. Cette dernière comporte des logiciels qui sont classés en deux catégories : les programmes d'application des utilisateurs et les programmes système qui permettent le fonctionnement de l'ordinateur. Parmi ceux-ci, le système d'exploitation(SE).

Un système d'exploitation est un ensemble de programmes qui contrôle et coordonne l'utilisation des ressources de l'ordinateur (mémoire, processeur, périphériques) par les différents utilisateurs et les différents logiciels applicatifs.

Le système d'exploitation est le logiciel qui prend en charge les fonctionnalités élémentaires du matériel et qui propose une plateforme plus efficace en vue de l'exécution des programmes. Il offre une large palette de fonctionnalités qui simplifient la création de logiciels applicatifs.

Le système d'exploitation est un logiciel central utilisé par tous les logiciels applicatifs pour exploiter le matériel de l'ordinateur, il est l'interface entre les programmes et le matériel.

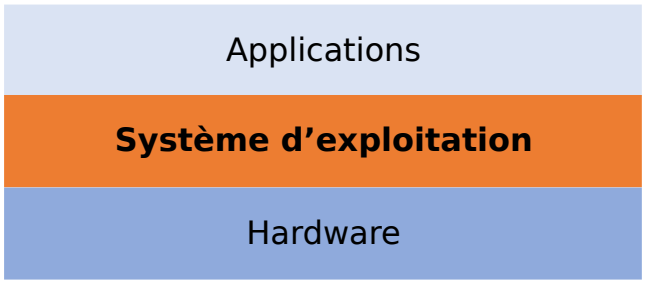


Figure 1 : Structure en couche basique d'un ordinateur

Dans le secteur informatique, les systèmes d'exploitation les plus répandus sont :

- **Windows** (pour les PC)
- **Mac OS** (pour les ordinateurs d'Apple)
- **Linux** (pour les PC et les serveurs)
- **Unix** (pour les serveurs).
- Pour les téléphones, on trouve **Android**, **iOS** (chez Apple), **Symbian** et **Windows Phone**.

1.3 Evolution des systèmes d'exploitations

Tout système d'exploitation dépend étroitement de l'architecture de l'ordinateur sur lequel il fonctionne.

Les premiers ordinateurs étaient mis à la disposition d'un programmeur selon un calendrier de réservation : un usager avec un travail unique utilisait seul la machine à un moment donné. Puis vint l'époque du traitement par lots (batch) : enchaînement, sous le contrôle d'un moniteur, d'une suite de travaux avec leurs données, confiés à l'équipe d'exploitation de la machine (inconvénient : temps d'attente des résultats pour chaque utilisateur).

1ère génération (1945-1955) : Tubes à vide et tableaux d'interrupteurs

- ***Moteurs de calcul à relais mécaniques*** : Très lents (temps de cycles mesurés en secondes)
- ***Tubes à vide*** *:* relais mécaniques remplacés par des tubes à vide, machines énormes remplissant des pièces entières (dizaines de milliers de tubes à vide).

La même équipe concevait, construisait, programmait, administrait et maintenait la machine

Tout programme était conçu en langage machine (les autres langages n'existaient pas)

Un programme était conçu en basculant des tableaux d'interrupteurs pour contrôler les fonctions de base de la machine (on programmait la machine avec un programme)

- Protocole classique d'utilisation de la machine

- Le programmeur demande une réservation de la machine pour une certaine durée
- Il insère son programme dans la machine en manipulant le tableau d'interrupteurs
- Dans les heures qui suivent, il prie pour qu'aucun des quelques 20.000 tubes ne grille pendant l'exécution

- Protocole amélioré avec les cartes perforées

- Un programme de calculs numériques simple est écrit sur des cartes, qui sont lues par la machine au lieu d'utiliser des tableaux d'interrupteurs pour 'programmer la machine'. Pas de système d'exploitation

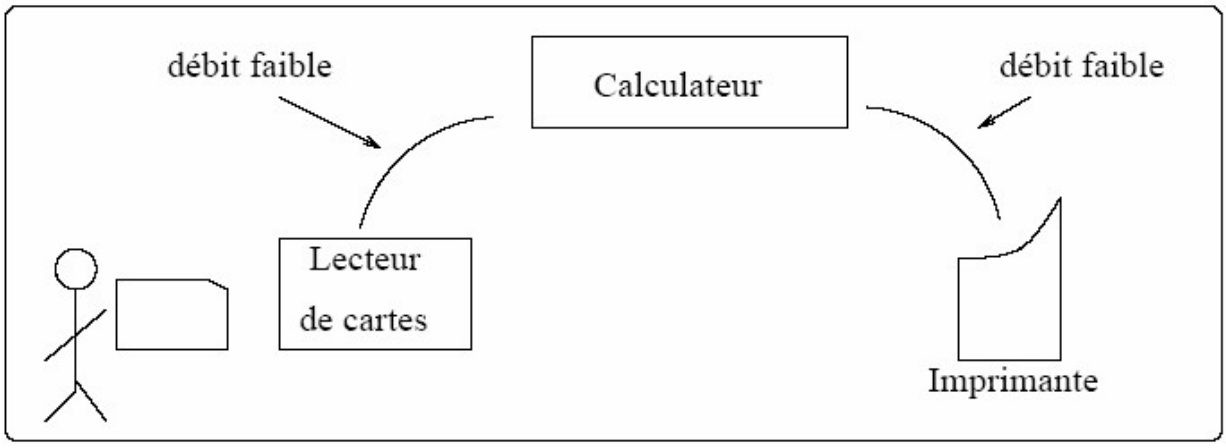


Figure 2 : Illustration (1ère génération)

2ème génération : (1955-1965) : Transistors et systèmes par lots

Le passage aux transistors rendait les ordinateurs plus fiables et permet la séparation entre concepteurs, constructeurs, programmeurs, opérateurs et personnel de maintenance**.** Les programmes étaient écrits en

FORTRAN puis codés sur des cartes perforées

- Protocole d'utilisation de la machine

- Le programmeur apporte son paquet de cartes dans la salle de soumission des *jobs* (tâches, programmes à exécuter)
- L'opérateur fait lire et exécuter les cartes par la machine
- L'opérateur récupère la trace d'exécution sur une imprimante et la stocke dans la salle des résultats pour que le programmeur la récupère.

- L'opérateur prend ensuite un autre paquet de cartes soumis et répète le processus précédent

- Si le compilateur FORTRAN est nécessaire, l'opérateur doit également le charger dans la machine

- ***Protocole lent :*** Un opérateur humain traite séquentiellement un job et gère la soumission des entrées

(lecture de cartes perforée) et des sorties (résultat sur imprimante) une fois le calcul terminé

- ***Amélioration du protocole :*** Insertion des cartes de contrôle de job pour réaliser cet enchaînement : découpage du traitement d'un job en plusieurs étapes (lecture des entrées, calcul, production des sorties) réalisées par des moniteurs de traitement par lot dit Moniteurs Batch Mono Job (MBMJ). Les entrées/sorties d'un job traité par une machine peu onéreuse et performante et le calcul est effectué par une machine performante (calculateur)

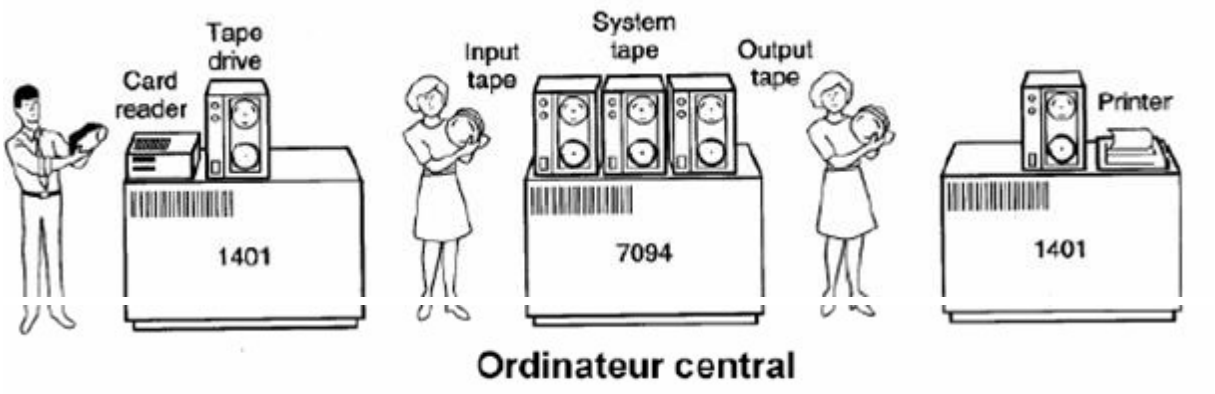


Figure 3 : **Illustration** ***(2ème génération)***

- ***Types de programmes :*** calculs scientifiques et d'ingénierie (ex. résolution d'équations aux dérivées partielles)
- ***Système d'exploitation :*** ce fut la naissance des systèmes d'exploitation tels que : FMS (*Fortran Monitor System*) et IBYS (1er système d'exploitation de l'IBM 7094)

3ème génération (1965-1980) : Circuits intégrés

Historiquement, on peut dire que les SE sont vraiment nés avec les ordinateurs de la 3ème génération (ordinateurs à circuits intégrés apparus après 1965). Le premier SE digne de ce nom est l'OS/360, celui des IBM 360, famille unique de machines compatibles entre elles, de puissances et de configurations différentes. Bien que son extrême complexité (due à l'erreur de couvrir toute la gamme 360) n'ait jamais permis d'en réduire le nombre de bogues, il apportait deux concepts nouveaux :

- ***Circuits intégrés :*** leur avantage au plan prix/performance a permet la création d'une seule gamme de produits avec la même architecture matérielle et le même jeu d'instructions et un même programme qui peut tourner sur toutes ces machines (pas de machine d'E/S et d'autres pour le calcul).
- ***Nouvelles techniques dans les systèmes d'exploitation***

- ***Multiprogrammation :*** charger plusieurs programmes dans la MC et de les exécuter en fonction des ressources qu'ils utilisent. Elle vise l'élimination des temps d'attente de l'UC pendant les opérations d'E/S. Entre autre, ce mécanisme de multiprogrammation a exigé l'apparition de la Mémoire partagée et les mécanismes de protection.
- ***Spool (Simultaneous Peripheral Operation Onl Line):*** transfert des travaux des cartes vers le disque permettant la lecture et l'écriture simultanées sur le disque et disparition de la manipulation des bandes

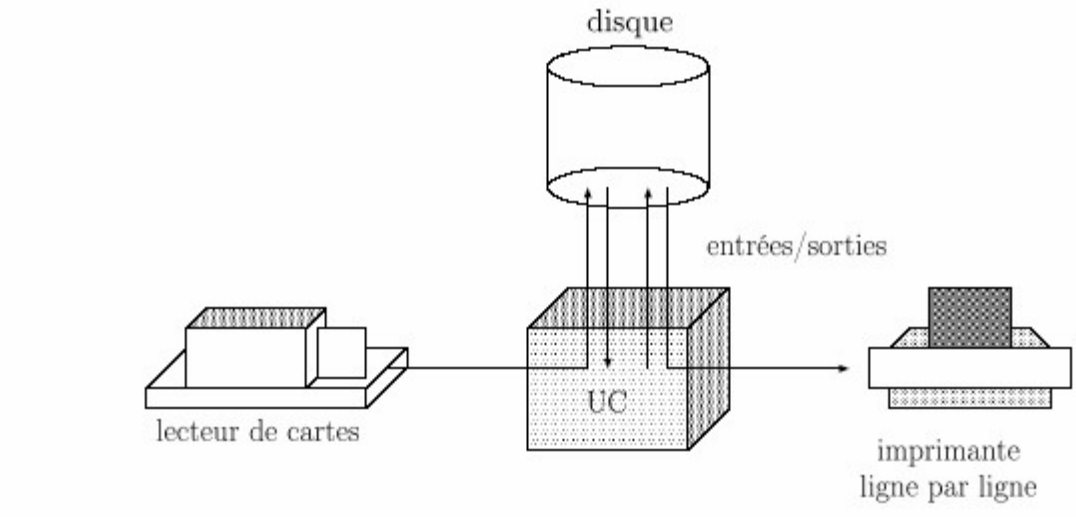


Figure 4 : Illustration (3ème génération) : Spooling

- ***Partage de temps*** : une variante de la multiprogrammation (chaque utilisateur possède un terminal en ligne) qui permet de partager le temps de l'UC entre les programmes qui résident dans la MC.

- **Exemples de systèmes d'exploitation**

- **MULTICS (**MULTiplexed Information and Computing Service)**:** système pour ordinateur central, capable de supporter des centaines d'utilisateurs simultanés
- **DEC et PDP-1, … PDP-11 :** Systèmes pour mini-ordinateurs ; performant pour des tâches non scientifiques ; de 4K mots de 18 bits, pour un prix de 120 000 $).

- **UNICS** (Uniplexed Information and Computer Service) : une version simplifiée (monoutilisateur) de

MULTICS qui allait devenir le célèbre UNIX écrit en langage C et le plus porté sur toutes sortes de machines

4ème génération (1980-aujourd'hui) : Ordinateurs personnels

- ***Circuit LSI (Large Scale Integration) :*** Circuits intégrés à haute densité contenant des milliers de transistors sur $1mm_2$ de silicium permettant le développement des micro-ordinateurs (ordinateurs personnels), très peu onéreux comparés aux mini-ordinateurs de type PDP-11
- ***Exemples de systèmes d'exploitation***
  - **CP/M (*Control Program for Microcomputers*)** : conçu pour Intel 8080, comprend un contrôleur pour le tout récent lecteur de disquettes 8 pouces
  - ***MS-DOS (MicroSoft-Disk Operating System): développer en 1980 pour l'IBM PC par B.***

Gates à partir du noyau DOS. Il intègre petit à petit des concepts riches d'UNIX et de MULTICS

  - **Macintosh d'Apple :** un SE qui intègre une IHM graphique, destiné à des utilisateurs qui ne connaissaient rien aux ordinateurs
  - **Windows de Microsoft :** successeur de MS-DOS qui intègre une IHM graphique influencé par le succès de

Macintosh (Windows 95/98/NT/2000/XP)

  - ***UNIX ; Linux : intègre de l'IHM graphique avec un système de fenêtres appelé X-Window***

1.4 Les rôles et composantes d'un système d'exploitation

1.4.1 Rôles et fonctionnalités du système d'exploitation

Un système d'exploitation (SE) est présent au cœur de l'ordinateur coordonnant les tâches essentielles à la bonne marche du matériel. C'est du système d'exploitation que dépend la qualité de la gestion des ressources (processeur, mémoire, périphériques) et la convivialité de l'utilisation d'un ordinateur.

1.4.1.1 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**. Le type d'ordonnanceur est totalement dépendant du système d'exploitation, en fonction de l'objectif visé.

1.4.1.2 Gestion de la mémoire vive

Sur une machine plusieurs programmes peuvent s'exécuter simultanément. Il ne faut pas que ceux-ci puissent interférer les uns avec les autres. Durant son déroulement, une application a besoin de sauvegarder des données temporairement. Quand vous utilisez un traitement de texte par exemple, il conserve la police actuellement utilisée. Cela se fait dans la mémoire. Et il ne faut pas qu'une autre application qui est en train d'être utilisé en même temps puisse altérer cette valeur.

Pour réaliser cela, les programmes ne peuvent pas écrire directement dans la RAM de la machine. Pour eux tout se passe comme si c'était le cas, mais le système d'exploitation se charge d'écrire dans certaines zones indépendantes les unes des autres. Le programme voit un espace de **mémoire virtuelle** dans lequel il peut faire ce qu'il veut sans risquer de déranger les autres.

Le système d'exploitation est chargé de gérer l'espace mémoire alloué à chaque application et, le cas échéant, à chaque usager. En cas d'insuffisance de mémoire physique, le SE peut créer une zone mémoire sur le disque dur, appelée «**mémoire virtuelle**» ou **swap**. 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 contrepartie cette mémoire est beaucoup plus lente. Un des premiers rôles du système d'exploitation est de gérer la mémoire disponible sur la machine. Par cela, il s'agit de la RAM présente sur la machine, mais pas seulement, comme cela sera expliqué ci-après.

Tout ce mécanisme est invisible pour l'application. C'est le système d'exploitation qui se charge de ces opérations selon les besoins. Les temps d'accès au disque dur sont coûteux. Donc si la mémoire est insuffisante par rapport aux applications utilisées, l'utilisation du swap sera intensive et ralentira la machine.

1.4.1.3 Gestion des entrées/sorties (périphériques)

Le SE permet d'unifier et de contrôler l'accès des programmes aux ressources matérielles par l'intermédiaire des pilotes (appelés également gestionnaires de périphériques ou gestionnaires d'entrée/sortie).

1.4.1.4 Gestion de l'exécution des applications (processus)

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.

1.4.1.5 Gestion des droits

Le SE est chargé de la sécurité liée à l'exécution des programmes en garantissant que les ressources ne sont utilisées que par les programmes et utilisateurs possédant les droits adéquats.

1.4.1.6 Gestion des fichiers

Le système d'exploitation gère la lecture et l'écriture dans le système de fichiers et les droits d'accès aux fichiers par les utilisateurs et les applications.

1.4.1.7 Gestion des informations

Le SE fournit un certain nombre d'indicateurs permettant de diagnostiquer le bon fonctionnement de la machine.

1.4.2 Les composantes d'un système d'exploitation (SE)

Un système d'exploitation est typiquement composé :

- ✓ d'un noyau
- ✓ d'un système de gestion de fichiers (File Management System)
- ✓ d'une interface utilisateur : un ensemble d'outils système (utilitaires – shell, graphique)

1.4.2.1 Noyau ou Kernel

Il assure les fonctionnalités suivantes :

- ✓ gestion des périphériques (au moyen de pilotes) ;
- ✓ Responsable de l'activation des composants électroniques et de l'interaction avec les programmes qui les gèrent.
- ✓ gestion des files d'exécution (aussi nommée processus) :
  - ➢ attribution de la mémoire à chaque processus ;
  - ➢ ordonnancement des processus (répartition du temps d'exécution sur le ou les processeurs).
  - ➢ synchronisation et communication entre processus (services de synchronisation, d'échange de messages, mise en commun de segments de mémoire, etc.)
- ✓ gestion des fichiers (au moyen de systèmes de fichiers) ;
- ✓ gestion des protocoles réseaux (TCP/IP, IPX, etc.).

1.4.2.2 Le système de gestion de fichier

Un **système de fichiers** (abrégé « **FS** » pour *File System*) ou **système de gestion de fichiers** (SGF) est une façon de stocker les informations et de les organiser dans des fichiers sur des mémoires secondaires (pour le matériel informatique, il s'agit de mémoire de masse comme un disque dur, un disque SSD, un CD-ROM, une clé USB, une disquette, etc.). Une telle gestion des fichiers permet de traiter, de conserver des quantités importantes de données ainsi que de les partager entre plusieurs programmes informatiques. Il offre à l'utilisateur une vue abstraite sur ses données et permet de les localiser à partir d'un chemin d'accès.

Le système de gestion de fichiers prend en charge les opérations de lecture-écriture de fichiers sur des ressources autres que la mémoire centrale (disque, disquette, clé USB...). Il gère à la fois les fichiers, les répertoires et tient à jour la table d'allocation (File Allocation Table) qui permet d'associer le fichier à son implantation physique sur le disque, mais également de gérer l'espace disponible sur celui-ci. Un certain nombre de commandes et d'utilitaires sont mises à la disposition de l'utilisateur pour gérer ses fichiers et ses répertoires (copie, renommage, création de répertoire, formatage...) FAT: petite zone qu'il y'a en début de disque dur qui permet au disque de savoir à quel fichier est rattaché le cluster (implantation sur le disque).

1.4.2.3 L'interface utilisateur

Cette partie permet d'échanger avec l'utilisateur ou avec les applications utilisateur, c'est la partie la plus externe, elle joue le rôle de passerelle entre l'utilisateur et le noyau. L'interface utilisateur peut être comparée à un interprète, traduisant les frappes de l'utilisateur, ses clics de souris ou toute autre entrée pour les programmes appropriés, elle comprend :

- ✓ Les utilitaires
- ✓ Le Shell
- ✓ L'interface graphique

Une **interface système** ou **coque logicielle** (***shell*** en anglais) est une couche logicielle qui fournit l'interface utilisateur d'un système d'exploitation. Il correspond à la couche la plus externe de ce dernier. L'interface système est utilisée comme diminutif de l'**interface utilisateur du système d'exploitation**.

De façon générale, il existe plusieurs types d'interfaces avec différentes fonctions et propriétés :

- L'**interface en ligne de commande** est un dispositif dans lequel l'utilisateur peut saisir des phrases correspondant aux opérations à effectuer.
- L'**interface graphique** est un dispositif dans lequel les objets à manipuler sont présentés sous forme de pictogrammes sur lesquels l'usager peut imiter des manipulations physiques avec une souris ;
- L'**interface naturelle**, est une interface de communication entre l'homme et la machine qui se doit d'être imperceptible. La communication se rapproche du langage humain.
- L'**interface système** ou ***shell*** en anglais, est un dispositif qui permet à l'utilisateur de commander un système d'exploitation.

1.5 Classification des systèmes d'exploitation

On peut classer les systèmes d'exploitation selon différents critères. On a ci-dessous un certains nombres de critères de classification.

1.5.1 Systèmes monotâche/multitâches

1.5.1.1 Systèmes monotâche

Les systèmes mono tâches sont caractérisés par un environnement où l'on ne peut exploiter qu'un seul programme à la fois. MS DOS, par exemple, est un système d'exploitation monotâche.

1.5.1.2 Système multitâche

Un système d'exploitation est dit multitâche » (en anglais *multithreaded*) lorsque plusieurs «**tâches**» (également appelées *processus*) peuvent être exécutées simultanément. Les applications sont composées en séquence d'instructions que l'on appelle «**processus légers**» (en anglais «*threads*»). Ces threads seront tour à tour actifs, en attente, suspendus ou détruits, suivant la priorité qui leur est associée ou bien exécutés séquentiellement.

Windows XP, 7, 8,10, LINUX (UNIX), etc. sont des systèmes multitâches.

Datacenter Server par exemple, peut supporter jusqu'à 32 processeurs. Lorsque les systèmes ne supportent pas les multiprocesseurs, le multitâche peut être simulé par un multitâche préemptif ou collaboratif.

Un système est dit **préemptif** lorsqu'il possède un **ordonnanceur** (aussi appelé *planificateur*), qui répartit, selon des critères de priorité, le temps machine entre les différents processus qui en font la demande.

Le système est dit à **temps partagé** lorsqu'un quota de temps est alloué à chaque processus par l'ordonnanceur. C'est notamment le cas des systèmes multi-utilisateurs qui permettent à plusieurs utilisateurs d'utiliser simultanément sur une même machine des applications différentes ou bien similaires : le système est alors dit «**système transactionnel**». Pour ce faire, le système alloue à chaque utilisateur une tranche de temps.

1.5.1.3 Multitâche préemptif

Lorsqu'un système fonctionne en mode multitâche préemptif, cela signifie qu'il est capable d'exécuter plusieurs processus ou applications en même temps sur une même machine. Le noyau du système interrompt les tâches les moins prioritaires, quand il le veut, au bout d'un certain temps ou si elles attendent une ressource non disponible.

Le passage d'une tâche à l'autre se fait tellement rapidement que l'on croit avoir un vrai multitâche.

1.5.1.4 Multitâche collaboratif ou temps partagé

Un ordinateur fonctionne en mode multitâche collaboratif, quand une tâche en cours d'exécution renonce volontairement au processeur à un moment donné, permettant à d'autres processus de s'exécuter.

Windows 3.1 est un exemple de système utilisant ce mode de fonctionnement.

1.5.2 Systèmes mono utilisateur / multi-utilisateurs

Un système mono utilisateur ne peut traiter qu'un seul utilisateur à un instant donné.

Les systèmes d'exploitation **multi utilisateurs** peuvent supporter plusieurs sessions en même temps.

Les caractéristiques principales sont :

- gestion d'environnement propre à chaque utilisateur (identification, ressources propres)
- sécurité d'accès aux programmes et aux données
- notion de droits d'accès

1.5.3 Les systèmes multiprocesseurs

Le **multiprocessing** est une technique consistant à faire fonctionner plusieurs processeurs en parallèle afin d'obtenir une puissance de calcul plus importante que celle obtenue avec un processeur haut de gamme ou bien afin d'augmenter la disponibilité du système (en cas de panne d'un processeur).

On appelle **SMP** (*Symmetric Multiprocessing* ou *Symmetric Multiprocessor*) une architecture dans laquelle tous les processeurs accèdent à un espace mémoire partagé.

Un système multiprocesseur doit donc être capable de gérer le partage de la mémoire entre plusieurs processeurs mais également de distribuer la charge de travail.

1.5.4 Les systèmes embarqués

Les **systèmes embarqués** sont des systèmes d'exploitation prévus pour fonctionner sur des machines de petite taille, telles que des téléphones, smart TV, PDA (*personal digital assistants* ou en français *assistants numériques personnels*) ou des appareils électroniques autonomes (sondes spatiales, robot, ordinateur de bord de véhicule, etc.), possédant une autonomie réduite. Ainsi, une caractéristique essentielle des systèmes embarqués est leur gestion avancée de l'énergie et leur capacité à fonctionner avec des ressources limitées. Les principaux systèmes d'exploitation «grand public» pour appareils embarqués sont les systèmes embarqués pour smartphone ou tablette tels qu'Android, IOS, Windows Phone, etc. Aujourd'hui les systèmes sont de plus en plus repandus et divers, surtout avec l'avènement de l'internet des objets ou « internet of things »

1.5.5 Les systèmes temps réel

Les **systèmes temps réel** (*real time systems*), essentiellement utilisés dans l'industrie, sont des systèmes dont l'objectif est de fonctionner dans un environnement contraint temporellement.

Un système temps réel doit ainsi fonctionner de manière fiable selon des contraintes temporelles spécifiques, c'est-à-dire qu'il doit être capable de délivrer un traitement correct des informations reçues à des intervalles de temps bien définis (réguliers ou non).

Voici quelques exemples de systèmes d'exploitation temps réel :

- OS-9 ;
- RTLinux (RealTime Linux) ;
- QNX ;
- VxWorks.

1.5.6 Les systèmes centralisés / repartis

Dans un système centralisé, l'ensemble des ressources utilisées par le système d'exploitation pendant les traitements se portent uniquement sur l'ordinateur où le système est installé.

Dans une architecture distribuée, l'ensemble des ressources (mémoire, processeur, etc.) utilisées par le système d'exploitation pendant les traitements sont repartis sur un ensemble d'ordinateurs en liaison ou en réseaux avec l'ordinateur où se trouve le système d'exploitation.

1.5.7 Les systèmes 32 bits / 64 bits

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. Le nombre de bits d'un système va de pair avec le nombre de bits du processeur. On peut installer un système sur un processeur de bits supérieurs, mais pas l'inverse.

Système	Codage	Mono-utilisateur	Multi-utilisateur	Mono-tâche	Multitâche
DOS	16 bits	X		X	
Windows3.1	16/32 bits	X			non préemptif
Windows95/98/Me	32 bits	X			coopératif
WindowsNT/2000	32 bits		X		préemptif
WindowsXP	32/64 bits		X		préemptif
Windows7	32/64 bits		X		préemptif
Unix / Linux	32/64		X		préemptif

	bits		
MAC/OS X	32 bits	X	préemptif
VMS	32 bits	X	préemptif

1.5.8 Les qualités d'un système d'exploitation

Les systèmes d'exploitation peuvent être jugés suivant différents critères :

- **La robustesse** (la protection de l'espace mémoire alloué à chaque processus)
- **La stabilité** (le système ne plante à tout bout de champ)
- **L'ouverture et la fiabilité** (le nombre d'application qui sont développées par les éditeurs indépendants pour fonctionner avec tel ou tel système.)
- **La rapidité** de traitement
- **L'interopérabilité** avec d'autres systèmes
- **La connectivité** réseau
- **L'approbation des utilisateurs**.
- **La fiabilité** : limiter les conséquences des défaillances matérielles ou des erreurs des utilisateurs. En cas de panne, éviter les pertes d'information ou leur incohérence.
- **Efficacité** : Utiliser au mieux les ressources et possibilités matérielles (sans en consommer trop pour lui-même).
- **Facilité d'emploi** : Offrir un langage de commande (dialogue usager/système) et des diagnostics d'erreurs (système/usager) clairs et précis.
- **Adaptabilité** : permettre des modifications matérielles et logicielles les plus simples possibles, à l'aide d'outils spécialisés.
- **Mesurabilité** : Enregistrer la comptabilité des ressources utilisées par les usagers, mesurer les paramètres de fonctionnement et de charge.

1.6 Structure d'un système d'exploitation

1.6.1 Structure d'un système informatique

L'architecture basique d'un système informatique est une architecture en 3 couches, comprenant une couche application, une couche système d'exploitation et une couche matérielle. Cette architecture peut être représentée comme suit :

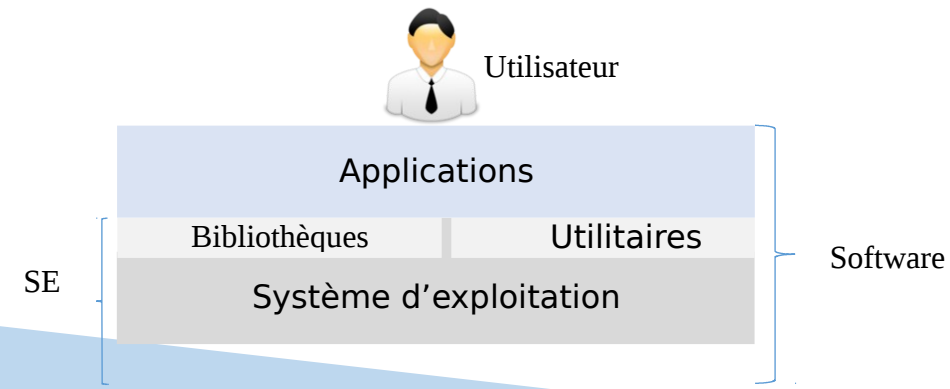


Figure 5 : Structure d'un système informatique

➢ Applications

C'est les logiciels avec lesquels l'utilisateur interagit directement. Une application en informatique représente un programme ou logiciel directement utilisé par l'utilisateur pour réaliser une tâche, ou un ensemble de tâches élémentaires d'un même domaine ou formant un tout. *Un éditeur de texte, un navigateur web, un lecteur multimédia, un jeu vidéo, etc. sont des applications*. Les applications s'exécutent en utilisant les services du système d'exploitation pour utiliser les ressources matérielles.

➢ Les utilitaires

Ce sont les services de base aux utilisateurs. Les utilitaires sont des logiciels conçus pour analyser, configurer, optimiser ou entretenir une pièce d'équipement informatique, un système d'exploitation, un logiciel ou les informations enregistrées sur un support informatique.

Un logiciel utilitaire est utilisé pour gérer un système informatique contrairement aux logiciels d'application, qui visent à exécuter des tâches qui profitent aux utilisateurs du système informatique.

Un certain nombre de logiciels utilitaires sont généralement livrés avec les systèmes d'exploitation. Cependant, ces programmes ne sont pas considérés comme faisant partie du système d'exploitation. Bien que les logiciels utilitaires livrés avec les systèmes d'exploitation soient de plus en plus complets et sophistiqués, les utilisateurs installent souvent des logiciels utilitaires tiers en remplacement ou en complément de ceux fournis avec le système d'exploitation.

Exemples :

- ***Les utilitaires antivirus*** : ils recherchent des logiciels malveillants et les suppriment.
- ***Les utilitaires d'archivage de fichiers*** : ils combinent un certain nombre de fichiers en un seul fichier d'archive ou une série de fichiers d'archives, pour en faciliter le transport ou le stockage. Ils peuvent inclure des capacités de compression et de cryptage.
- ***Les utilitaires de sauvegarde*** : ils enregistrent des copies des informations stockées sur un fichier, un dossier ou un disque et rétablissent soit des fichiers sélectionnés (par exemple, en cas de suppression accidentelle), soit le disque entier (par exemple, en cas de bris du disque).
- ***Les utilitaires de presse-papiers*** : ils étendent la fonctionnalité du presse-papiers d'un système d'exploitation.
- ***Les utilitaires cryptographiques*** : ils chiffrent et déchiffrent les flux et les fichiers.
- Les utilitaires de compression de données : ils convertissent une suite de bits en une suite plus courte.

- ***Les utilitaires de génération de données*** : ils créent un fichier de données de test à partir de critères spécifiés.
- ***Les utilitaires de synchronisation de données*** : ils maintiennent la cohérence entre deux ou plusieurs sources. Ils peuvent être utilisés pour créer des copies de redondance ou de sauvegarde, mais sont également utilisés pour aider les utilisateurs à gérer leurs musiques, leurs photos et leurs vidéos dans leurs appareils.
- ***Les utilitaires de contrôle de versions*** : ils conservent plusieurs versions d'un ou de plusieurs fichiers et permettent de comparer, restaurer ou fusionner des versions.
- ***Les utilitaires de vérification de disque*** : ils balaient un disque dur en recherchant et en tentant de corriger les erreurs logiques (erreur du système de fichiers) ou physiques qui s'y trouvent.
- ***Les utilitaires de nettoyage de disque*** : ils identifient des fichiers qui ne sont pas nécessaires au fonctionnement de l'ordinateur et occupent de l'espace inutilement.
- ***Les utilitaires de compression de disque*** : ils compressent et décompressent de manière transparente le contenu d'un disque, augmentant ainsi sa capacité.
- ***Les utilitaires de défragmentation de disque*** : ils détectent les fichiers dont le contenu est dispersé sur plusieurs emplacements sur le disque dur et déplace les fragments en un seul emplacement pour augmenter l'efficacité des accès au fichier.
- ***Les utilitaires de partition de disque dur*** : ils divisent un disque physique en plusieurs disques logiques, chacun avec son propre système de fichiers qui peut être traité par le système d'exploitation comme un disque individuel.
- ***Les utilitaires d'analyse d'espace disque*** : ils fournissent une visualisation de l'utilisation de l'espace disque par les dossiers, les sous-dossiers et les fichiers du système.
- ***Les utilitaires de gestion de fichiers*** : ils effectuent des tâches de gestions de fichiers et de dossiers comme la création, le renommage, le déplacement, la suppression, la restauration, la copie, la fusion de fichiers et de dossiers.
- ***Les utilitaires d'édition hexadécimale*** (éditeur hexadécimal) : ils modifient directement le contenu d'un fichier. Les fichiers peuvent être des fichiers de données ou des programmes.
- ***Les utilitaires de test de mémoire*** : ils vérifient le bon fonctionnement des éléments de mémoire vive.
- ***Les utilitaires réseaux*** : ils analysent la connexion de l'ordinateur au réseau, configurent les paramètres réseau, vérifient le transfert de données et enregistrent les événements relatifs au réseau.
- ***Les utilitaires de gestion de programmes*** (ou gestionnaire de paquets) : ils automatisent le processus d'installation, de configuration, de mise à jour et de désinstallation de logiciels installés sur un système informatique.
- ***Les utilitaires de nettoyage de registre*** : ils nettoient et optimisent la base de registre Windows en supprimant les anciennes clés de registre qui ne sont plus utilisées.
- ***Les utilitaires d'écran de veille*** : originalement, les écrans de veille étaient utilisés pour prévenir la combustion interne du phosphore des écrans d'ordinateur de vieille technologie. La combustion interne du phosphore n'est plus un problème sur les écrans modernes. Aujourd'hui, les écrans de veille sont utilisés principalement pour le divertissement ou la sécurité.
- ***Les utilitaires de tri et de fusion*** : ils organisent des enregistrements dans un fichier dans une séquence spécifiée.
- ***Les utilitaires de monitorage de système*** : ils permettent de voir en temps réel l'état et la performance d'un système informatique et de ses composantes.

Remarque

On peut opposer deux grands types de logiciels :

- les **logiciels de base**, chargés d'assurer l'exploitation d'un ordinateur (c'est-à-dire son fonctionnement interne), comme :
  - le système d'exploitation,
  - les utilitaires, qui facilitent et sécurisent l'exploitation de l'ordinateur (tels un antivirus, des logiciels de sauvegarde, d'installation ou de désinstallation, de compression de fichiers) ;
  - les pilotes pour interagir avec le matériel informatique.
- les **logiciels d'application**, qui traitent chacun à leur façon l'information, les données, permettant ainsi à l'utilisateur de produire et de lire des documents dans des domaines très divers (par exemple un traitement de texte, un tableur ou un navigateur).

➢ Les Bibliothèques

Les bibliothèques mettent à disposition du système d'exploitation et des programmes applicatifs des morceaux de programmes tout prêts, dont le but est de faciliter l'accès à certaines fonctions. Grâce aux bibliothèques (.dll, .OCX, ...), les développeurs peuvent facilement et rapidement réutiliser des fonctions utiles, sans avoir à les reprogrammer eux-mêmes.

➢ Le système d'exploitation

C'est le premier logiciel de base qu'on installe sur un ordinateur et qui permet d'exploiter les ressources de celui-ci. Un **système d'exploitation** (souvent appelé **OS** - de l'anglais *Operating System*) est un ensemble de programmes qui dirige l'utilisation des ressources d'un ordinateur par des logiciels applicatifs. Il reçoit des demandes d'utilisation des ressources de l'ordinateur — ressources de stockage des mémoires et des disques durs, ressources de calcul du processeur, ressources de communication vers des périphériques ou via le réseau — de la part des logiciels applicatifs.

Le SE est chargé d'assurer la liaison entre les ressources matérielles, l'utilisateur et les applications. Ainsi, lorsqu'un programme doit accéder à une ressource matérielle, il ne lui est pas nécessaire d'envoyer ces informations au système qui se charge de les transmettre au périphérique concerné par l'intermédiaire de son pilote. En l'absence de pilote, il faudrait que chaque programme reconnaisse et prenne en compte la communication avec chaque type de périphérique. Le SE permet ainsi de dissocier les programmes et le matériel afin notamment de simplifier la gestion des ressources et offrir à l'utilisateur une IHM simplifiée afin de lui permettre de s'affranchir de la complexité de la machine physique.

➢ Les drivers ou pilotes

Driver est un mot anglais qui peut signifier **« pilote »** ou « conducteur », c'est un programme permettant à un système d'exploitation de reconnaître un matériel et de l'utiliser ; un driver permet de piloter un matériel informatique ou d'interagir avec lui.

Les drivers sont considérés comme faisant partie du système d'exploitation. Ils ne le sont pas forcément au sens logiciel, car ils peuvent être fournis séparément par le constructeur. Mais ils s'exécutent dans un mode spécial, appelé **mode réel**. Ce mode leur permet de contourner les protections du système d'exploitation du **mode protégé** (dont notamment le contrôle de la mémoire).

Tous les périphériques disposent de drivers (logiciels pilotes). Les pilotes concernent tout aussi bien les périphériques internes (carte graphique, carte réseau, carte son, chipset contrôleur…) qu'externes (imprimante, scanner, webcam, APN…). Les drivers sont fournis avec le matériel ou disponibles sur le site du constructeur. Certains OS intègrent déjà une base des drivers les plus répandus.

➢ Le matériel

Il représente l'ensemble des composants matériels d'un ordinateur tels que le processeur, la RAM, la ROM, le disque dur, etc.

Du hardware au software, nous avons de façon encore plus détaillée cette représentation :

1. Assistance à l'utilisateur : agents intelligents
2. Interface utilisateur : vocale, métaphore du bureau, icônes, souris, etc.
3. Logiciel d'application : tableur, texteur, ludiciel, butineur, bases de données, etc.
4. Environnement de développement : AGL, débogueur, etc.
5. Logiciel de développement : compilateurs, éditeurs, SGBD, etc.
6. Logiciel système utilisateur : TS, spool, etc.
7. Langage de commande du système et du réseau : shell, JCL, etc.
8. Langage de requêtes au système : SVC, macros
9. Gestionnaire de ressources, comptabilité, etc.
10. Gestionnaire de mémoire virtuelle : pagination, segmentation
11. Gestionnaire des fichiers (SGF, FMS)
12. Gestionnaire des entrées sorties (IOCS, BIOS)
13. Ordonnanceur des tâches (scheduler)
14. Synchronisation des tâches (primitives P et V sur sémaphores, etc.)
15. Langage machine

16. Entrées sorties physiques
17. Interruptions (matérielles et logicielles)
18. Microcode : langage lié au matériel (registres, portes, etc.)
19. Architecture de l'ordinateur (bus, ram, rom, puces, etc.)
20. Architecture logique des puces (portes, bus, etc.)
21. Architecture physique des puces : VLSI...
22. Support physique des puces : couches MOS
23. Matériaux des circuits : silicium, etc.

1.6.2 La structure en couche d'un système d'exploitation

Les couches permettent d'identifier les niveaux d'interdépendance entre les fonctions fournies par le SE.

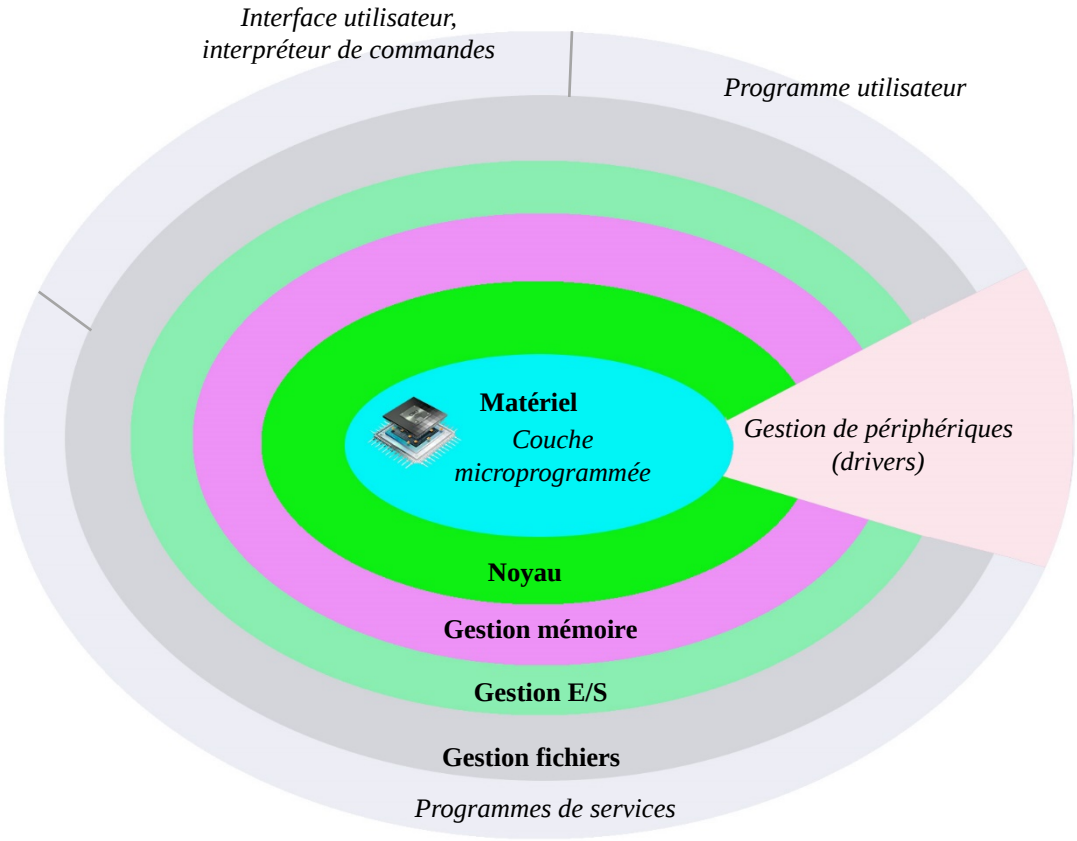


Figure 6 : Structure en couche d'un SE moderne

➢ **Le noyau**

- Réside en mémoire (fréquence élevée des interventions)
- Petite taille
- Gestion du processeur: reposant sur un allocateur (*dispatcher*) responsable de la répartition du temps processeur entre les différents processus, et un planificateur (*scheduler*) déterminant les processus à
- activer, en fonction du contexte.
- Gestion des *interruptions* : les *interruptions* sont des signaux envoyés par le matériel, à destination du logiciel, pour signaler un évènement.
- Gestion du multi- tâches: simuler la simultanéité des processus coopératifs (i. e. les processus devant se synchroniser pour échanger des données) et gérer les accès concurrents aux ressources
- (fichiers, imprimantes, ...)

➢ **Le gestionnaire de mémoire**

La mémoire centrale a toujours été une ressource critique: initialement très coûteuse et peu performante (tores magnétiques), elle était de très faible capacité.

Le gestionnaire de la mémoire qui se charge du partage de la mémoire entre les processus en exécution.

➢ **Le système de gestion de fichiers**

Le concept de fichiers est une structure adaptée aux mémoires secondaires et auxiliaires permettant de regrouper des données. Le rôle d'un système d'exploitation est de donner corps au concept de fichiers (les gérer, c'est- à- dire les créer, les détruire, les écrire (modifier) et les lires, en offrant la possibilité de les désigner par des noms symboliques).

➢ **Les Entrées/ Sorties**

Il s'agit de permettre le dialogue (échange d'informations) avec l'extérieur du système.

La tâche est rendue ardue, par la diversité des périphériques d'entrées- sorties et les multiples méthodes décodage des informations (différentes représentations des nombres, des lettres, etc.)

Concrètement, la gestion des E/S implique que le SE mette à disposition de l'utilisateur des procédures standard pour l'émission et la réception des données, et qu'il offre des traitements appropriés aux multiples conditions d'erreurs susceptibles de se produire (plus de papier, erreur de disque, débit trop différent, ...)

➢ **L'invite des commandes ou shell**

Nécessaire pour interagir avec l'utilisateur, il peut être

- Graphique
- Console interpréteur de commandes (langage de commande interprété).

Il attend les ordres que l'utilisateur transmet par le biais de l'interface, décode et décompose ces ordres en actions élémentaires, et finalement réalise ces actions en utilisant les services des couches plus profondes du système d'exploitation.

Outre l'interaction «directe» (au moyen de terminaux ou de consoles dans le cas d'Unix *ou MS DOS*), les systèmes s'offrent le «*traitement par lots*» *(batch*). Ce mode de traitement non- interactif est obtenu en regroupant les commandes dans un fichier alors appelé *script*.

1.6.3 Les différents types de noyaux

Il existe toutes sortes de noyaux, plus ou moins spécialisés. Des noyaux spécifiques à une architecture, souvent monotâches, d'autres généralistes et souvent multitâches et multi-utilisateurs. 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. 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).

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.

1.6.3.1 Noyaux monolithiques non modulaires

Certains systèmes d'exploitation, comme d'anciennes versions de Linux, certains BSD ou certains vieux Unix ont un noyau monolithique. C'est-à-dire que l'ensemble des fonctions du système et des pilotes sont regroupés dans un seul bloc de code et un seul bloc binaire généré à la compilation.

De par la simplicité de leur concept mais également de leur excellente vitesse d'exécution, les noyaux monolithiques ont été les premiers à être développés et mis en œuvre. Cependant, au fur et à mesure de leurs développements, le code de ces noyaux monolithiques a augmenté en taille et il s'est avéré difficile de les maintenir. Le support par les architectures monolithiques des chargements à chaud ou dynamiques implique une augmentation du nombre de pilotes matériels compilés dans le noyau, et par suite, une augmentation de la taille de l'empreinte mémoire des noyaux. Celle-ci devient rapidement inacceptable. Les multiples dépendances créées entre les différentes fonctions du noyau empêchaient la relecture et la compréhension du code. L'évolution du code s'est faite en parallèle à l'évolution du matériel, et des problèmes de portage ont alors été mis en évidence sur les noyaux monolithiques.

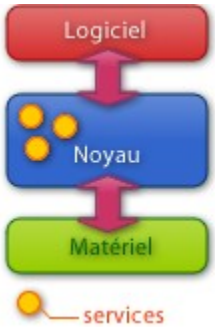


Figure 7 : Architecture monolithique.

En réalité les problèmes de la portabilité de code se sont révélés avec le temps indépendants de la problématique de la technologie des noyaux. Pour preuve, NetBSD est un noyau monolithique et est porté sur un très grand nombre d'architectures, alors que des noyaux tels que GNU Mach ou NT utilisent des micro-noyaux censés faciliter le portage mais n'existent que pour quelques architectures.

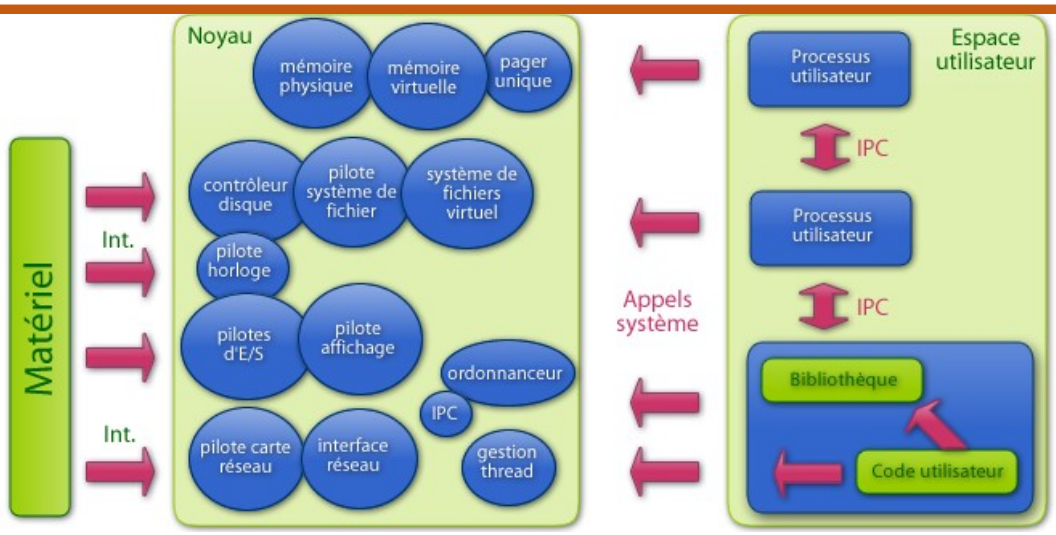


Figure 8 : Architecture d'un noyau monolithique.

1.6.3.2 Noyaux monolithiques modulaires

Pour résoudre les problèmes évoqués ci-dessus, les noyaux monolithiques sont devenus modulaires. Dans ce type de noyau, 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 qui peuvent être séparés tant du point de vue du code que du point de vue binaire.

La très grande majorité des systèmes actuels utilise cette technologie : Linux, la plupart des BSD ou Solaris. Par exemple avec le noyau Linux, certaines parties peuvent être non compilées ou compilées en tant que modules chargeables directement dans le noyau. La modularité du noyau permet le chargement à la demande de fonctionnalités et augmente les possibilités de configuration. Ainsi les systèmes de fichiers peuvent être chargés de manière indépendante, un pilote de périphérique changé, etc. Les distributions Linux, par exemple, tirent profit des modules chargeables lors de l'installation. L'ensemble des pilotes matériels sont compilés en tant que modules. Le noyau peut alors supporter l'immense variété de matériel trouvé dans les compatibles PC. Après l'installation, lors du démarrage du système, seuls les pilotes correspondants au matériel *effectivement* présent dans la machine sont chargés en mémoire vive. La mémoire est économisée.

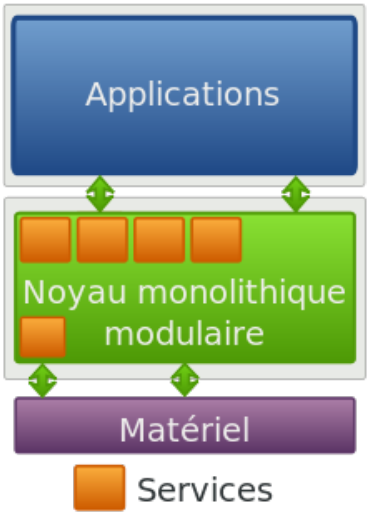


Figure 9 : Architecture d'un système à noyau monolithique modulaire.

Les noyaux monolithiques modulaires conservent les principaux atouts des noyaux monolithiques purs dont ils sont issus. Ainsi, la facilité de conception et de développement est globalement maintenue et la vitesse d'exécution reste excellente. L'utilisation de modules implique le découpage du code source du noyau en blocs indépendants. Ces blocs améliorent l'organisation et la clarté du code source et en facilitent également la maintenance.

Les noyaux monolithiques modulaires conservent également un important défaut des noyaux monolithiques purs : une erreur dans un module met en danger la stabilité de tout le système. Les tests et certifications de ces composants doivent être plus poussés.

D'un point de vue théorique, le grand nombre de lignes de code exécutées en mode noyau engendre des problèmes de portabilité. La pratique contredit largement la théorie et les noyaux modulaires sont aujourd'hui les plus portés.

1.6.3.3 Systèmes à micro-noyaux

1.6.3.3.1 fonctionnement

Les limitations des noyaux monolithiques ont amené à une approche radicalement différente de la notion de noyau : les systèmes à 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.

Un petit nombre de fonctions fondamentales est conservé dans un noyau minimaliste appelé « micronoyau ». L'ensemble des fonctionnalités habituellement proposées par les noyaux monolithiques est alors assuré par les services déplacés en espace utilisateur et par ce micro-noyau. Cet ensemble logiciel est appelé « micronoyau enrichi ».

Ce principe a de grands avantages théoriques : en éloignant les services « à risque » des parties critiques du système d'exploitation regroupées dans le 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.

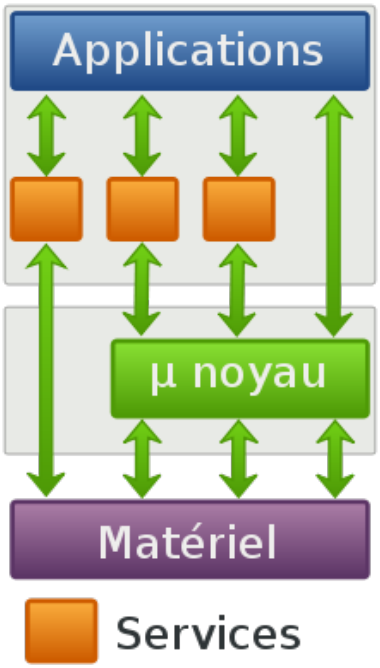


Figure 10 : Architecture d'un système à micro-noyau.

1.6.3.3.2 Avantages et inconvénients d'un système à micro-noyau

Les avantages théoriques des systèmes à micro-noyaux sont la conséquence de l'utilisation du mode protégé par les services qui accompagnent le micro-noyau. En effet, en plaçant les services dans l'espace utilisateur, ceux-ci bénéficient de la protection de la mémoire. La stabilité de l'ensemble en est améliorée : une erreur d'un service en mode protégé a peu de conséquences sur la stabilité de l'ensemble de la machine.

De plus, en réduisant les possibilités pour les services de pouvoir intervenir directement sur le matériel, la sécurité du système est renforcée. Le système gagne également en possibilités de configuration. Ainsi, seuls les services utiles doivent être réellement lancés au démarrage. Les interdépendances entre les différents services sont faibles. L'ajout ou le retrait d'un service ne perturbe pas l'ensemble du système. La complexité de l'ensemble est réduite.

Le développement d'un système à micro-noyau se trouve également simplifié en tirant parti à la fois de la protection de la mémoire et de la faible interdépendance entre les services. Les erreurs provoquées par les applications en mode utilisateur sont traitées plus simplement que dans le mode noyau et ne mettent pas en péril la stabilité globale du système. L'intervention sur une fonctionnalité défectueuse consiste à arrêter l'ancien service puis à lancer le nouveau, sans devoir redémarrer toute la machine.

Les micro-noyaux ont un autre avantage : ils sont beaucoup plus compacts que les noyaux monolithiques. 6 millions de lignes de code pour le noyau Linux 2.6.0 contre en général moins de 50 000 lignes pour les micro-noyaux. La maintenance du code exécuté en mode noyau est donc simplifiée. Le nombre réduit de lignes de code peut augmenter la portabilité du système.

Les premiers micro-noyaux (comme Mach) n'ont pas tout de suite atteint ces avantages théoriques. L'utilisation de nombreux services dans l'espace utilisateur engendre les deux problèmes suivants :

1. La plupart des services sont à l'extérieur du noyau et génèrent un très grand nombre d'appels système ;
2. Les interfaces de communication entre les services (IPC) sont complexes et trop lourdes en temps de traitement.

Le grand nombre d'appels système et la communication sous-jacente sont un défaut inhérent à la conception des micro-noyaux. Dans L4, il a été résolu en plaçant encore plus de services en espace utilisateur. La rapidité de traitement des IPC a pu être améliorée en simplifiant les communications au maximum, par exemple en supprimant toute vérification des permissions, laissant ce soin aux serveurs externes.

Ces modifications radicales ont permis d'obtenir de bonnes performances mais elles ne doivent pas faire oublier qu'un micro-noyau doit être accompagné d'un grand nombre de services pour fournir des fonctionnalités équivalentes à celles des noyaux monolithiques. De plus, la grande liberté dont disposent les services au niveau de la sécurité et de la gestion de la mémoire accroît la difficulté et le temps de leur développement (ils doivent fournir leurs propres interfaces).

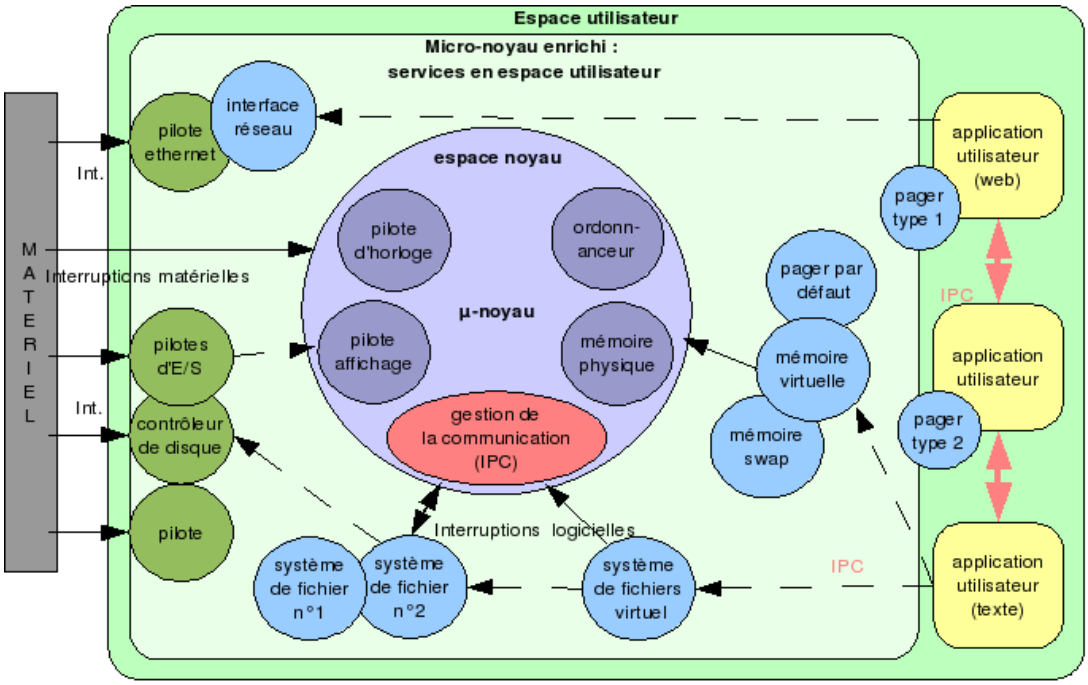


Figure 11 : Architecture d'un micro-noyau enrichi par des services (micro-noyau enrichi).

1.6.3.4 Noyaux hybrides

La dénomination « noyaux hybrides » désigne principalement des noyaux qui reprennent des concepts à la fois des noyaux monolithiques et des micro-noyaux, pour combiner les avantages des deux.

Lorsque, au début des années 1990, les développeurs et concepteurs se sont aperçus des faiblesses des premiers micro-noyaux, certains réintégrèrent diverses fonctionnalités non fondamentales dans le noyau, pour gagner en performance. Les micro-noyaux « purs » semblaient condamnés à l'échec.

Alors que la philosophie générale des systèmes à micro-noyaux est maintenue (seules les fonctions fondamentales sont dans l'espace noyau), certaines fonctions non critiques, mais très génératrices d'appels système, sont réintégrées dans l'espace noyau. Ce compromis permet d'améliorer considérablement les performances en conservant de nombreuses propriétés des systèmes à micro-noyaux. Un exemple de ce type de noyau hybride est le noyau XNU de Mac OS X. Il est basé sur le micro-noyau Mach 3.0, mais qui inclut du code du noyau monolithique BSD au sein de l'espace noyau.

Cette dénomination est également utilisée pour désigner d'autres types de noyaux, notamment les noyaux monolithiques sur micro-noyaux (temps réel ou non) tels que L4Linux (Linux sur L4), MkLinux (le noyau Linux sur Mach), Adeos, RTLinux et RTAI.

Plus rarement, on peut rencontrer le terme « noyau hybride » pour remplacer improprement « noyau monolithique modulaire » ou « micro-noyau enrichi ».

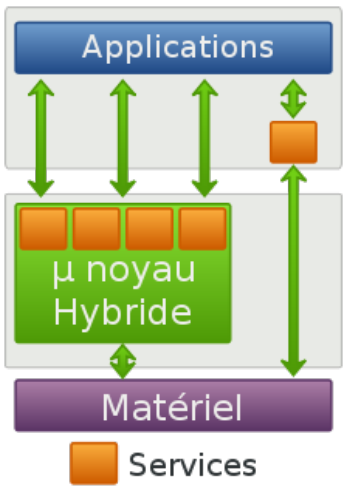


Figure 12 : Architecture hybride.

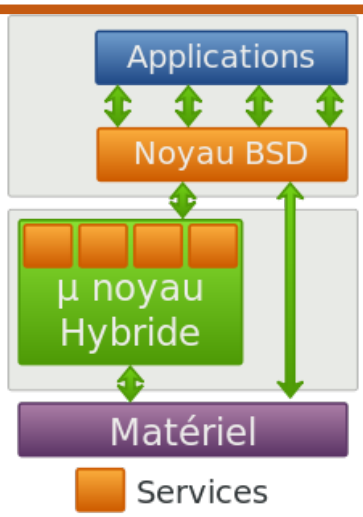


Figure 13 : Architecture hybride : *XNU.*

1.6.3.5 Exo-noyaux

Étymologiquement, 'exo' signifie en grec 'hors de'. Un exo-noyau est donc un système d'exploitation fonctionnant en espace utilisateur (en 'user-space', au lieu du 'kernel-space' dans le cas des autres noyaux). Les fonctions et services du système d'exploitation sont assurés par de petits modules qui, selon les approches techniques, sont des bibliothèques dynamiques (MIT, LibOSes) ou des démons (IntraServices).

1.6.3.6 Méta-noyaux

Un « méta-noyau » est un ensemble de logiciels qui vise à appliquer la notion de noyau informatique au niveau d'un réseau informatique, en créant une unique couche de gestion des périphériques au niveau d'un réseau.

De cette manière, les logiciels peuvent être déployés et utilisés sur le réseau informatique comme s'il s'agissait d'une machine unique, et l'ensemble des logiciels fonctionnant sur cette plate-forme peuvent se partager les ressources de manière intégrée, comme elle le ferait sur un noyau simple.

Un méta système doit également permettre la personnalisation, la gestion des permissions ainsi que l'utilisation d'informations dépendant de la localisation.

Cette notion rejoint les notions de grappe de calcul, de machine virtuelle, de serveur d'applications et de CORBA.

1.6.3.7 Noyaux temps réel

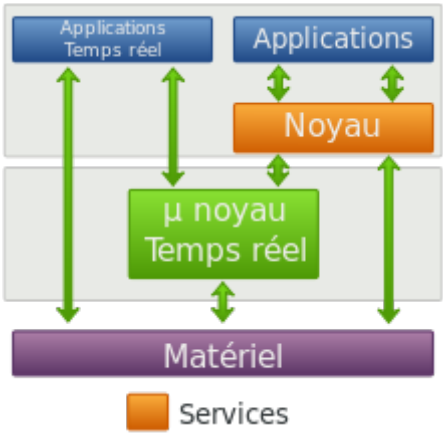


Figure 14 : Une possibilité d'architecture de noyau temps réel hybride.

Les noyaux temps réel sont fonctionnellement spécialisés. Ce sont des noyaux généralement assez légers qui ont pour fonction de base stricte de garantir les temps d'exécution des tâches. Il n'y a pas à proprement parler de notion de rapidité de traitement ou de réactivité dans les noyaux temps réel, cette notion est plutôt implicite à la garantie des temps d'exécution en comparaison aux critères temporels de l'application industrielle (la réactivité d'un système de freinage ABS n'a pas les mêmes critères temporels que le remplissage d'une cuve de pétrole).

Très utilisés dans le monde de l'électronique embarquée, ils sont conçus pour tourner sur des plates-formes matérielles limitées en taille, puissance ou autonomie.

Les noyaux temps réel peuvent adopter en théorie n'importe quelle architecture précédemment listée. Ils fournissent souvent deux interfaces séparées, l'une spécialisée dans le temps réel et l'autre générique. Les applications temps réel font alors appel à la partie temps réel du noyau.

Une des architectures souvent retenue est un noyau hybride qui s'appuie sur la combinaison d'un micro-noyau temps réel spécialisé, allouant du temps d'exécution à un noyau de système d'exploitation non spécialisé. Le système d'exploitation non spécialisé fonctionne en tant que service du micro-noyau temps réel. Cette solution permet d'assurer le fonctionnement temps réel des applications, tout en maintenant la compatibilité avec des environnements préexistants.

Par exemple, on peut avoir un micro-noyau temps réel allouant des ressources à un noyau non temps réel tel que Linux (RTLinux, RTAI, Xenomai) ou Windows (RTX). L'environnement GNU (resp. Windows) peut alors être exécuté à l'identique sur le noyau pour lequel il a été conçu, alors que les applications temps réel peuvent faire directement appel au micro-noyau temps réel pour garantir leurs délais d'exécutions.

VxWorks est un noyau propriétaire temps réel très implanté dans l'industrie bien que les systèmes à base de noyau Linux se déploient énormément et aient un succès grandissant via RTAI et Xenomai (RTLinux étant breveté).

1.7 La virtualisation

1.7.1 Définition

La virtualisation est un mécanisme informatique qui consiste à faire fonctionner plusieurs systèmes, serveurs ou applications, sur un même serveur physique. La virtualisation est un composant technique clé dans le Cloud Computing.

1.7.2 Comment ça marche ?

La virtualisation repose sur le mécanisme suivant :

- Un système d'exploitation principal (appelé « système hôte ») est installé sur un serveur physique unique. Ce système sert d'accueil à d'autres systèmes d'exploitation.
- Un logiciel de virtualisation (appelé « hyperviseur ») est installé sur le système d'exploitation principal. Il permet la création d'environnements clos et indépendants sur lesquels seront installés d'autres systèmes d'exploitation (« systèmes invités »). Ces environnements sont des « machines virtuelles ».
- Un système invité est installé dans une machine virtuelle qui fonctionne indépendamment des autres systèmes invités dans d'autres machines virtuelles. Chaque machine virtuelle dispose d'un accès aux ressources du serveur physique (mémoire, espace disque…).

1.7.3 Usages

La virtualisation permet différents types d'applications :

- installation de plusieurs systèmes d'exploitation sur un unique serveur,
- mise en place d'un Plan de retour d'activité rapide en cas d'incident,
- test des applications sur plusieurs systèmes dans les phases de développement,
- accélération de la montée en puissance du système d'information.

1.7.4 Avantages

La virtualisation offre les avantages suivants :

- consolidation et rationalisation d'un parc de serveurs en entreprise : les entreprises ne sont plus obligées d'acheter un serveur physique pour chaque application,
- rationalisation des coûts de matériels informatiques,
- possibilité d'installer plusieurs systèmes (Windows, Linux) sur une même machine,
- portabilité des serveurs : une machine virtuelle peut être déplacée d'un serveur physique vers un autre (lorsque celle-ci a, par exemple, besoin de davantage de ressources),
- accélération des déploiements de systèmes et d'applications en entreprise,
- administration simplifiée de l'ensemble des serveurs,
- réduction de la facture d'électricité, en diminuant le nombre de serveurs physiques.

1.7.5 Inconvénients

Quelques inconvénients existent autour de la virtualisation :

- coût important : pour faire fonctionner convenablement une architecture virtualisée, l'entreprise doit investir dans un serveur physique disposant de plusieurs processeurs et de beaucoup de mémoire,
- pannes généralisées : si le serveur physique tombe en panne, les machines virtuelles tombent également en panne,
- vulnérabilité généralisée : si l'hyperviseur est bogué ou exposé à une faille de sécurité, les machines virtuelles peuvent l'être également et ne sont plus protégées. La virtualisation, en augmentant les couches logicielles, a pour conséquence d'augmenter la surface d'attaque de l'entreprise.

1.7.6 Les logiciels de virtualisation

Les logiciels les plus répandus sont :

- VirtualBox
- VMware Workstation
- Parallels desktop
- Windows Virtual PC
- Vmlite

1.8 Liste des systèmes d'exploitation

1.8.1 Les principaux

- **Microsoft Windows** (dernière version : Windows 10) : les systèmes d'exploitation de Microsoft sont actuellement préinstallés sur plus de 90 % des ordinateurs personnels. Chaque version de windows se decline en plusieurs familles (Windows Home, Windows Starter, Windows RT, Windows familiale, Windows Professionnel, Windows integrale, etc.)
- Dérivés d'**Unix** (sous différentes déclinaisons : BSD, System V, etc.) dont :
    - **Mac OS X**: système préinstallé sur la majorité des ordinateurs vendus par Apple ;
    - **GNU/Linux** : un système d'exploitation libre s'appuyant sur le noyau Linux et les outils GNU installés sur + de 1 % du parc informatique mondial toutes distributions confondues[réf. nécessaire].
        - Systèmes **GNU/Linux pères**: ArchLinux, Debian, Gentoo, Red Hat, SUSE, Slackware, etc.

        - Systèmes **GNU/Linux fils** : OpenSUSE, Fedora, Mandriva, Ubuntu, Linux Mint, Manjaro, PCLinuxOS, Centos, Kali Linux, etc.

    - la famille **BSD** : un effort réussi pour rendre sa liberté au système de Berkeley comprenant :
        - NetBSD, OpenBSD et son dérivé OliveBSD, FreeBSD et ses dérivés PicoBSD, DragonFly BSD et PC-BSD; Darwin (sur lequel est construit Mac OS X, semi-propriétaire), OpenSolaris de Sun.
    - les Unix dits « propriétaires » :
        - AIX (IBM, SystemV), A/UX (Apple, SystemV), BOS (Bull Operating System), IRIX (Silicon Graphics, SystemV), HP-UX (Hewlett-Packard, SystemV), LynxOS (LynuxWorks), NeXTSTEP (NeXT, BSD), Sinix (Siemens), Solaris (Sun, SystemV), SunOS (Sun, BSD), Tru64 (Compaq).

- **Mac OS** : le premier système d'exploitation des ordinateurs Macintosh d'Apple, qui a succédé aux systèmes Lisa et Apple II, et a été remplacé par Mac OS X ;
- **OS/2 d'IBM** et son successeur eComStation ;
- **OS/400** présent sur les moyens systèmes IBM (AS/400 - ISéries) ;
- **VMS** et **OpenVMS** (Supporté par HP qui a acheté Compaq, ex-Digital) ;
- les systèmes d'exploitation ***mainframes*** (« grands systèmes ») :
    - Multics (père d'UNIX) et héritier de CTSS
    - IBM : MVS, VM, DOS/VSE, TPF
    - Bull : GCOS
    - Siemens : BS2000
    - ITS, TOPS-10 et TOPS-20
    - PipinOS ; Basse sous BeOS developer en C par UmiXTY Computer Software Inc.

1.8.2 Systèmes d'exploitation pour smartphone

- **Android**, système d'exploitation développé par Google avec un noyau Linux. Il occupe en 2016 86,2% de parts de marché (contre 12,9% pour iOS)
- **iOS** (anciennement iPhone OS), développé par Apple
- **BlackBerry OS**, développé par BlackBerry
- **Windows Phone**, développé par Microsoft

- **Symbian**, développé par Nokia
- **MeeGo**, développé par Nokia et Intel (noyau Linux)
- **Palm OS**, développé par Palm
- **Bada**, développé par Samsung
- TangoOS, prochaine génération de système d'exploration pour Smartphone par Kerple Computer (Linux)
- **Tizen**, développé en partie par Samsung (pour ne plus être dépendant à Android, donc à Google)
- **Firefox OS**, développé par la Mozilla Foundation
- **Ubuntu Touch**, développé par Canonical (noyau Linux)
- **HP webOS,** développé par Palm
- **CopperheadOS**, développé par Copperhead basé sur Android avec des modifications pour la sécurité (noyau GNU/Linux)
- **Sailfish OS**, développé par Jolla

1.8.3 Système d'exploitation pour TV

- **Android TV**, développé par Google
- **Tizen**, développé par Samsung
- **tvOS**, développé par Apple
- **Firefox OS**, développé par la Mozilla Foundation, choisi par Panasonic
- **WebOS**, repris par LG (anciennement par Palm)

1.8.4 Autres systèmes d'exploitation

1. **AcidOS**, système d'exploitation français écrit en C et assembleur, destiné à sécuriser et fiabiliser les systèmes informatiques industriels du serveur à l'IoT;
2. **AmigaOS**, le système d'exploitation des Amiga ;
3. **BeOS, Be**, la société qui le produisait a été rachetée par Palm qui elle-même a été rachetée par HP, le système n'est plus maintenu. voir HaikuOS ;
4. **FreeDOS, DOS** compatible avec MS-DOS et PC-DOS open source et gratuit ;
5. **Google Chrome OS**, système basé sur le navigateur Google Chrome avec un noyau Linux ;

6. **HaikuOS**, projet visant à réécrire entièrement BeOS sous une licence libre.
7. Illumos, un fork d'OpenSolaris visant à supprimer toutes les parties d'OpenSolaris dont le code n'est pas considéré comme libre et dont le but affiché est de garder une ABI 100% compatible avec OS/NET Solaris / OpenSolaris.;
8. iRMX, un système d'exploitation multitâche temps réel par Intel ;
9. Isaac, un système d'exploitation écrit en Lisaac ;
10. Jolicloud, un système d'exploitation basé sur Linux, simplifié et destiné à une utilisation Web sur le Cloud ;
11. KerpleOS,Systeme d'explotation de la Famille Windows NT ecrit en C++.II est comatible avec Windows et ReactOS (Developer par Umixty Computer)
12. Lepton, un système d'exploitation open source POSIX dédié aux systèmes embarqués temps réels ;
13. LynxOS, système d'exploitation temps réel style Unix pour systèmes embarqués et logiciels critiques ;
14. Minix, clone d'Unix basé sur un micro-noyau créé par Andrew S. Tanenbaum à des fins pédagogiques ;
15. MiNT, noyau multitâche inspiré des systèmes Unix BSD pour ordinateurs compatibles TOS. La compatibilité avec les anciennes applications est conservée (dans une certaine mesure) ;
16. NetWare, système d'exploitation uniquement serveur, édité par Novell. Première plateforme ayant hébergé le méta-annuaire Novell eDirectory (sous le nom de NDS à l'époque, pour Novell Directory Services). Novell a cessé son support en mars 2010, invitant à migrer vers Novell Open Enterprise Server basé sur SUSE Linux Enterprise Server ;
17. ReactOS, projet libre visant à une compatibilité des pilotes et des logiciels avec les différentes versions de Microsoft Windows de Famille NT. Logiciel libre sous GNU GPL, GNU LGPL et Licence BSD ;
18. RedHawk, OS temps réel, basé sur Linux, de Concurrent Computer ;
19. Red Star OS est un système d'exploitation nord-coréen basé sur le noyau Linux. Il est développé depuis 2002 par le Korea Computer Center.
20. SkyOS, un système d'exploitation propriétaire pour PC ;
21. Smaky, un système d'exploitation en français rendu libre en 2008 ;
22. SmartOS, basé sur OpenSolaris / Illumos avec de nombreuses modifications comme l'inclusion de la technologie de virtualisation du noyau Linux, KVM. Ce système d'exploitation est conçu pour l'hébergement de services dans le Cloud;
23. TRON, système d'exploitation japonais ;

24. VxWorks, un système d'exploitation temps réel de la firme Wind River Systems racheté le 3 juin 2009 par Intel. VXworks est employé par la NASA pour les missions spatiales Mars Pathfinder, Stardust, ainsi que pour les deux *rovers* martiens Spirit et Opportunity. Il est aussi employé pour gérer les pacemakers ou encore sur certains missiles ;

1.8.5 Caractéristiques techniques

OS	Architectures possibles	Système de fichiers possible	Type de noyau	Environnement graphique intégré	API
AIX	POWER, PowerPC	JFS, JFS2, ISO 9660, UDF, NFS, SMBFS, GPFS	Micro-noyau	Non	SysV, POSIX
FreeBSD	Intel IA32 (x86), AMD64, PC98, SPARC, autres	UFS2, ext2, FAT, ISO 9660, UDF, NFS, ZFS, autres	Monolithique avec des modules	Non	BSD, POSIX
HP-UX	PA-RISC,IA-64	CFS, HFS, ISO 9660, NFS, SMBFS, UDF, VxFS	Monolithique avec des modules	Non	SysV, POSIX
GNU/Linux	Presque toutes	Presque tous	Monolithique avec des modules	Non (sauf avec X Window, très répandu)	
Inferno	Intel IA32 (x86), Alpha, MIPS, PowerPC, SPARC, autres	Styx/9P2000, kfs, FAT, ISO 9660	Monolithique avec des modules	Oui	Propriétaire
Mac OS	PowerPC, 68k	HFS+, HFS, ISO 9660, FAT, UDF	Monolithique avec des modules	Oui	Propriétaire, Carbon
Mac OS X	PowerPC, Intel IA32 (x86)	HFS+ (defaut), UFS, AFP, ISO 9660, FAT, UDF, NFS, SMBFS, NTFS (lecture seulement)	Hybride	Oui	Carbon, Cocoa, BSD/POSIX, X11 (depuis la 10.3)
NetBSD	Intel IA32 (x86), 68k, Alpha, AMD64, PowerPC, SPARC, playstation2, dreamcast (60 plateformes)	UFS, UFS2, ext2, FAT, ISO 9660, NFS, LFS, autres	Monolithique avec des modules	Non	BSD, POSIX
NetWare	Intel IA32 (x86)	NSS, NWFS, FAT, NFS, AFP, UDF, ISO 9660	Hybride	Non	Propriétaire
OpenBSD	Intel IA32 (x86), 68k, Alpha, AMD64, SPARC, VAX, autres	UFS, ext2, FAT, ISO 9660, NFS, quelques autres	Monolithique avec des modules	Non	BSD, POSIX
OpenVMS	VAX, Alpha, IA-64	Files-11, ISO 9660, NFS	Monolithique avec des modules	Non	Unix-like
OS/2	Intel IA32 (x86)	HPFS, JFS, FAT, ISO 9660, UDF, NFS	Monolithique avec des modules	Oui	Propriétaire
Plan 9	Intel IA32 (x86), Alpha, MIPS, PowerPC, SPARC, autres	fossil/venti, 9P2000, kfs, ext2, FAT, ISO 9660	Monolithique avec des modules	Oui	Unix-like (et optionnellement POSIX)
Solaris	SPARC, SPARC64, AMD64, Intel IA32 (x86)	UFS, ZFS, ext2, FAT, ISO 9660, UDF, NFS, quelques autres	Monolithique avec des modules	Non	SysV, POSIX
Windows Server 2003	Intel IA32 (x86), AMD64, IA-64	NTFS, FAT, ISO 9660, UDF	Hybride	Oui	Win32, Win64
Windows XP	Intel IA32 (x86), AMD64, IA-64	NTFS, FAT, ISO 9660, UDF	Hybride	Oui	Win32, Win64
Windows Me	Intel IA32 (x86)	FAT, ISO 9660, UDF	Monolithique avec des modules	Oui	Win32
Haïku OS	Intel IA32 (x86), PowerPC, (AMD64 en développement)	BFS (défaut), FAT, ISO 9660, UDF, HFS, AFP, ext2, CIFS, NTFS (lecture seulement), ReiserFS (lecture seulement)	Hybride	Oui	POSIX, BeOS API
	Architectures possibles	**Système de fichiers possible**	**Type de noyau**	**Environnement graphique intégré**	**API**