Système d’exploitation

Dernière mise à jour : 24/01/2024

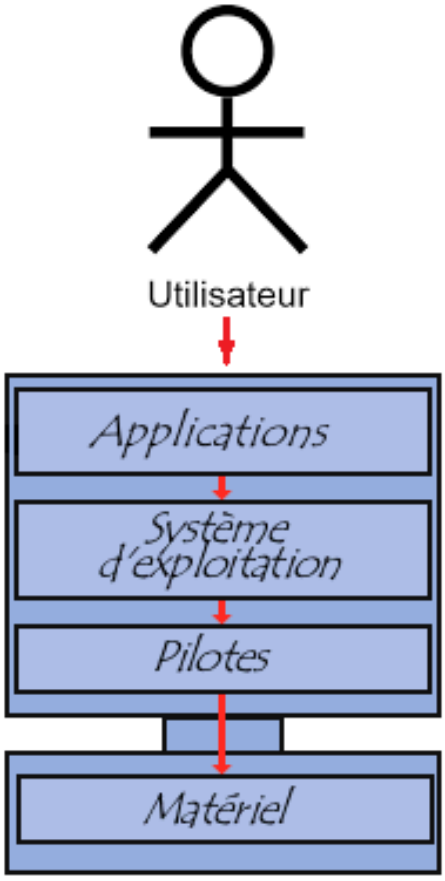
Table des matières

I. Introduction

Un **système d'exploitation (OS : Operating System)** est un **ensemble de programmes qui dirige l'utilisation des ressources d'un ordinateur** par des logiciels applicatifs.

Une **application** (**applicatif, appli**, **app)** est un **programme** (ou un ensemble logiciel) directement utilisé pour **réaliser une tâche**, ou un ensemble de tâches élémentaires d'un même domaine ou formant un tout. Exemples : éditeur de texte, navigateur web, lecteur multimédia, jeu vidéo, etc... Les applications s'exécutent en utilisant les services du système d'exploitation pour utiliser les ressources matérielles.

Le **système d'exploitation** (**OS**), est donc chargé d'assurer la liaison entre les ressources matérielles, l'utilisateur et les applications. Lorsqu'un **programme** désire **accéder à une ressource** matérielle, il ne lui est pas nécessaire d'envoyer des informations spécifiques au périphérique, il lui suffit d'**envoyer les informations au système d'exploitation**, qui se charge de les **transmettre au périphérique concerné via son pilote**. En l'absence de pilotes il faudrait que chaque programme reconnaisse et prenne en compte la communication avec chaque type de périphérique. On pourra ainsi représenter le processus global avec le schéma suivant :



Les **demandes d'utilisation des ressources de l'appareil** permettent d'éviter les interférences ou conflits entre les différents logiciels.

Le **système d'exploitation** reçoit des demandes d'utilisation des ressources de l'ordinateur de la part des logiciels applicatifs :

- **ressources de stockage des mémoires** (par exemple des accès à la mémoire vive, aux disques durs)

- **ressources de calcul du processeur central**,

- **ressources de communication vers des périphériques** (pour parfois **demander des ressources de calcul au GPU** par exemple ou tout autre carte d'extension) ou via le réseau

ll existe des pilotes pour quasiment tous les composants de votre ordinateur, smartphone, etc... :

- Carte graphique,
- Processeur,
- Disques durs ou SSD
- Mémoire (par exemple la mémoire RAM)
- Périphériques : clavier / souris, écran, imprimante, etc...

Ceux-ci permettent de communiquer avec les applications, programmes ou logiciels (par exemple : traitement de texte, antivirus, lecteur multimédia, applications du smartphone (YouTube, Chrome, etc…)

Les **pilotes** servent à faire le lien, ou à **traduire**, les **instructions** que vous envoyez à votre ordinateur afin que les composants exécutent l'action demandée.

Tout le fonctionnement de votre appareil dépend donc de son système d'exploitation. C'est pour cela que la majorité des appareils vendus disposent déjà d'un OS préinstallé.

Dès que vous allumez votre appareil, le système d'exploitation est, après le **firmware** et le **programme d'amorçage**, le premier système à être lancé. Une fois démarré, Il dispose dans la majorité des cas d'une interface graphique qui vous permet d'utiliser votre appareil de façon pratique, ludique et ergonomique.

Le **démarrage d'un ordinateur** ***(*****boot, initial program load : IPL*****)*** est la procédure de démarrage d'un ordinateur et comporte notamment le chargement du programme initial (**amorçage,** bootstrap).

On distingue :

- le « démarrage à froid » **(cold boot),** obtenu en allumant la machine, ou en l'éteignant puis en la rallumant ;

- le « démarrage à chaud » **(warm boot,** réamorçage, redémarrage, reboot), obtenu en **rechargeant le programme initial** ; il ne s'agit **pas d'un redémarrage au sens strict** (pas de coupure puis remise de l'alimentation électrique) ;

Un **firmware** (**micrologiciel**, **microcode**, **logiciel interne**, **logiciel embarqué** ou encore **microprogramme**) est un **programme intégré dans un matériel informatique** (ordinateur, photocopieur, automate (API, APS), disque dur, routeur, appareil photo numérique, etc...) pour qu'il puisse fonctionner.

Le firmware permet à un **matériel informatique d'évoluer** (via des mises à jour), d'**intégrer de nouvelles fonctionnalités**, sans avoir besoin de revoir complètement le design du hardware.

La mémoire dans laquelle réside le firmware pourra être :

- **non volatile :** stocke le programme et les données même lorsqu'elle n'est pas alimentée en électricité (ROM, EPROM, mémoire flash)
- **volatile :** effacée lorsqu'elle n'est plus alimentée en électricité. Dans ce cas, le firmware doit être chargé par un pilote à la mise sous tension, ce qui est peu pratique.

Dans la plupart des cas, ce logiciel gère le **fonctionnement interne du système électronique**.

Le firmware cumule les avantages du **logiciel**, qu'il est possible de **modifier**, et du **matériel**, plus **efficace**. Le firmware interagit avec des composants matériels qui ne peuvent plus être modifiés une fois fabriqués, ce qui réduit la nécessité de le mettre à jour.

Le **firmware des ordinateurs (BIOS ou EFI**) est exécuté par son (ou ses) CPU interne (exemples : code de gestion de la carte mère, d'une carte vidéo ou SCSI) ;

Le **firmware interne à chaque périphérique** est exécuté par le matériel que celui-ci contient (exemples : lecteur DVD, IDE ou SCSI, employant des micro-contrôleurs ou circuits de diverses familles).

Les deux cas précédents sont parfois réunis, par exemple, lorsque la ROM d'un périphérique contient une partie du BIOS de l'ordinateur ainsi que le firmware destiné à la gestion de ce périphérique. Cela permet de mesurer le degré d'indépendance d'un périphérique vis-à-vis du type de machine hôte car t**oute extension munie d'un BIOS pour PC n'est utilisable que sur une machine de ce type** alors que celles qui n'emploient qu'un **firmware** sont **indépendantes** de la **machine hôte**. C'est par exemple le cas des lecteurs/graveurs CD/DVD IDE, lesquels sont conformes à la norme ATAPI et fonctionnent donc sur toute machine IDE (PC, Mac…).

Le **BIOS (Basic Input Output System,** système élémentaire d'entrée/sortie) est un **ensemble de fonctions, contenu dans la mémoire morte (*ROM*)** de la carte mère d'un ordinateur, lui permettant d'effectuer des **opérations de base**, lors de sa mise sous tension. Par exemple l'identification des périphériques d'entrée/sortie connectés et la lecture d'un secteur sur un disque, un CD ou une partie d'une clé USB. Par extension, le terme est souvent utilisé pour décrire l'ensemble du micrologiciel de la carte mère. C'est en quelque sorte le **centre de contrôle de la carte mère**. Sur les cartes récentes il est remplacé par sa version moderne **l'UEFI.**

Un système d'exploitation apporte **commodité, efficacité et capacité d'évolution**, permettant d'**introduire de nouvelles fonctions** et du **nouveau matériel** sans remettre en cause les logiciels.

Il existe sur le marché des dizaines de systèmes d'exploitation différents, très souvent livrés avec l'appareil informatique. C'est le cas de Windows, Mac OS, Symbian OS, GNU/Linux, (pour lequel il existe de nombreuses distributions) ou Android. Les fonctionnalités offertes diffèrent d'un système à l'autre et sont typiquement en rapport avec l'exécution des programmes, l'utilisation de la mémoire centrale ou des périphériques, la manipulation des systèmes de fichiers, la communication, ou la détection et la gestion d'erreurs. Toutefois, la modélisation **CIM Schema** attribue à ce concept une classe de base **CIM_OperatingSystem,** éventuellement dérivée sous Windows, Linux ou z/OS (développé pour les ordinateurs centraux 64 bits zSeries par IBM en 2001. C'est le successeur du système OS/390. z/OS est une combinaison de MVS et d'UNIX System Services).

CIM Schema est une spécification informatique qui fait partie du Common Information Model, créé par le Distributed Management Task Force.

C'est un **schéma conceptuel** fait de classes, d'attributs, de relations entre ces classes et d'héritages, définis dans le monde du logiciel (**Software**) et du matériel informatique (**Hardware**). Cet ensemble d'objets et les relations qui les unissent représente une **base commune pour décrire des éléments informatique**s.

En 2012, les deux familles de systèmes d'exploitation les plus populaires sont Unix (dont macOS, GNU/Linux, iOS et Android) et Windows. Cette dernière détient un quasi-monopole sur les ordinateurs personnels avec près de 90 % de part de marché depuis plus de 20 ans.

https://www.ginjfo.com/actualites/logiciels/windows-10/windows-10-sa-part-de-marche-passe-la-barre-des-60-20200902

Windows 10	60.57%
Windows 7	22.31%
Mac OS X 10.15	5.03%
Windows 8.1	2.69%
Linux	2.33%
Mac OS X 10.14	1.96%
Mac OS X 10.13	1.26%
Windows XP	0.78%
Mac OS X 10.12	0.54%
Windows 8	0.53%

II. Windows 11

Avec les versions récentes de Windows, des modifications apparaissent d'un système d'exploitation à un autre, d'autres au contraire, ne changent pas (gestionnaire de périphérique vu précédemment, mais aussi le panneau de configuration qui sert encore aujourd'hui à faire des manipulations bien spécifiques.

Exemple :

1) Pas de changement :

- Action à définir lors de la fermeture de l'écran du pc portable : toujours via le panneau de configuration → Système

Panneau de configuration\Matériel et audio\Options d'alimentation\Paramètres système

2) Changements :

- Menu déroulant avec le clic droit de la souris (possibilité de revenir à l'ancien avec une ligne de commande pour modifier la base de registre. A lancer à chaque mise à jour ???)

- Onglets dans l'explorateur de fichiers

- Impression d'écran : Avant - « Windows + Shift + S ». Maintenant – touche « IMP ECR »

Et vous, avez-vous des exemples à donner de changement ou non depuis votre mise à jour vers Windows 11 ?

Date de fin de mise à jour de Windows 10 ?

III. Historique

Il n’y a pas toujours eu de système d’exploitation sur les ordinateurs. Les premiers ordinateurs fonctionnaient avec des programmes qui **communiquaient sans intermédiaire avec leurs ressources matérielles**. Ces ordinateurs étaient peu puissants, volumineux et mono tâches (cf machine de turing créée par Alan Turing).

Ainsi, les premières générations d'ordinateurs, dans les années 1945 à 1955, ne comportaient pas de système d'exploitation. Dans ces ordinateurs équipés de **tubes à vide**, les programmes manipulaient les ressources matérielles de l'ordinateur sans passer par un intermédiaire.

Un **tube électronique** (thermionic valve, vacuum tube), également appelé **tube à vide** ou même **lampe**, est un composant électronique actif, généralement utilisé comme **amplificateur de signal**. Le tube à vide redresseur ou amplificateur a été remplacé dans beaucoup d'applications par différents **semi-conducteurs**, mais n'a pas été remplacé dans certains domaines comme l'amplification de forte puissance ou des hyperfréquences.

L'ordinateur était utilisé par une seule personne à la fois : la tâche de l'opérateur consistait à placer des piles de cartes perforées dans le lecteur, où chaque carte comportait des instructions d'un programme ou des données. Les ordinateurs à tube à vide de cette génération n'avaient qu'une faible puissance de calcul, ils étaient volumineux, peu commodes et peu fiables (les tubes à vide grillaient souvent).

Il faut attendre les **années 1960** pour que les **premiers systèmes d’exploitation** apparaissent, bien qu’ils soient encore rudimentaires. Ils permettaient de placer en mémoire et de lancer plusieurs programmes simultanément.

Avec l'arrivée des circuits électroniques à semi-conducteurs, la puissance de calcul des processeurs a augmenté de manière significative. Les ordinateurs ont été équipés d'un **spooler**, une **file d'attente**, on utilise le terme notamment pour un serveur d’impression : spooler d’impression listant la

liste de tâches envoyées à l'imprimante par exemple. Il permet d'utiliser la puissance de calcul du processeur pendant que l'opérateur introduit les cartes. L'utilisation des ressources matérielles par les programmes se faisaient alors par l'intermédiaire d'une bibliothèque logicielle. Il a alors été possible de placer en mémoire plusieurs programmes simultanément et de les exécuter simultanément ; un programme dit **resident monitor** résidait continuellement dans la mémoire centrale et contrôlait l'exécution des différents programmes.

En **1965** le Massachusetts Institute of Technology (MIT) se lance dans la création du **premier système d'exploitation multitâche et multi-utilisateurs** : Multics (pour *MULTiplexed Information and Computing Service*, service multiplexé d'information et de calcul). Sur le principe de la multiprogrammation, le système d'exploitation autorisait le chargement de plusieurs programmes en mémoire et gérait le passage de l'un à l'autre, mais cette fois-ci sans attendre le blocage d'un programme. Chaque programme était exécuté pendant une durée de quelques millisecondes, puis le système passait au suivant. Ce temps, très court, donnait l'illusion que les programmes s'exécutaient simultanément. C'est d'ailleurs un principe de fonctionnement et une illusion qui existe encore avec les systèmes d'exploitation contemporains.

De plus, ces programmes pouvaient appartenir à des **utilisateurs distincts**, chacun ayant l'impression que la machine travaille uniquement pour lui. La possibilité pour un ordinateur de **servir simultanément plusieurs personnes** augmentait le retour sur investissement de l'achat de matériel très coûteux par les entreprises et les institutions. Cependant, du fait de son écriture dans un langage de programmation PL/I trop complexe pour les ordinateurs de l'époque, Multics fut un échec commercial.

PL/I ou **PL/1** (**Programming Language number 1**, Langage de programmation numéro 1) est un langage de programmation développé par IBM dans les débuts des années 1960.

Son objectif était d'être **universel** et de pouvoir remplacer indifféremment les langages à destination scientifique, tels que FORTRAN, ALGOL et COBOL, plus adapté aux problèmes de comptabilité et de gestion. Il permettait même l'accès à des fonctions autrefois réservées à la programmation système, comme la gestion de zones dynamiques de mémoire allouées à la demande (et non simplement à l'entrée dans un bloc), de pointeurs, et le travail par programme directement dans les tampons d'entrée-sortie.

En **1969**, les ingénieurs Ken Thompson et Dennis Ritchie des laboratoires Bell rêvent d'utiliser le système d'exploitation Multics, mais le matériel pour le faire fonctionner est encore hors de prix. Thompson se lance dans l'écriture d'une **version allégée de Multics** pour un PDP-7 inutilisé. Le système, fonctionnel, est surnommé **Unics** (***UNiplexed Information and Computing Service*T**), 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.

Dès **1980**, les **circuits électroniques à transistor ont été remplacés par des circuits intégrés**, plus petits, ce qui a permis de réaliser des appareils plus compacts et moins coûteux et lancé le marché des ordinateurs personnels. De nombreux concepteurs de système d'exploitation qui se sont lancés sur ce marché n'avaient pas d'expérience, ce qui a donné de nouveaux produits, fondés sur des nouvelles idées, sans héritage ou influence de ce qui se faisait jusqu'alors. **CP/M** créé par Gary Kildall et mis sur le marché en **1974**, a été le **premier système d'exploitation pour micro-ordinateur**, son caractère très sympathique, facile à aborder et commode (user-friendly) l'a rendu très populaire et influencé le marché des systèmes d'exploitation.

En **1980**, IBM prend contact avec Bill Gates, cofondateur 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 Bill Gates leur conseille 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**.

Xerox, une des sociétés majeures de l'époque, s'intéresse à l'optique de Steve Jobs. Elle réunit une poignée de scientifiques et d'ingénieurs dans son centre de recherche de Palo Alto et développe le premier micro-ordinateur équipé d'une interface utilisateur graphique, sur la base de thèses et d'études en ergonomie effectuées les années précédentes. Le résultat de ces recherches, le Xerox Star, ne sera jamais commercialisé. Dix ans plus tard, c'est Apple avec le Macintosh qui popularise les recherches effectuées par Xerox.

(cf film sur l'entreprise Apple et son fondateur Steve Jobs : Jobs en 2013)

En **1983**, Richard Stallman du Massachusetts Institute of Technology (MIT) lance l'idée d'un système d'exploitation sous licence libre : **GNU**. Il développe des outils de programmation, des logiciels utilitaires, et crée la **GNU General Public License.** C'est un contrat de licence autorisant une utilisation sans restrictions ainsi que la publication du code source, sa modification, et sa redistribution. Le succès est immédiat, mais le système ne possède toujours pas, en 1990, de noyau libre, et les tentatives pour en développer un sont loin d'être abouties.

En **1987**, Andrew Tanenbaum, professeur à l'université libre d'Amsterdam créé le système d'exploitation **Minix, clone d'UNI**X dont le code source est destiné à illustrer son cours sur la construction des systèmes d'exploitation. Mais Minix, dont la vocation est pédagogique, comporte alors de trop nombreuses limitations techniques et ne permet pas une utilisation poussée.

En **1989** un **système d'exploitation libre** apparaît à la même époque : **BSD**. La **Berkeley Software Distribution (BSD)** est la version d'UNIX développée par les étudiants et les chercheurs de l'université de Berkeley depuis 1977. Les logiciels utilitaires, créés sous licence libre, sont vendus avec le noyau Unix de AT&T, lui-même sous licence propriétaire. **AT&T** est aujourd'hui le plus grand fournisseur de services téléphoniques locaux et longues distances et de xDSL des États-Unis et le 2^{e} opérateur de services mobiles. **Digital subscriber line**, **DSL** (**xDSL,** ligne d'accès numérique,

ligne numérique d'abonné) renvoie à l'ensemble des techniques mises en place pour un transport numérique de l'information sur une ligne de raccordement filaire téléphonique ou liaisons spécialisées.

Cette double licence de BSD est à l'origine de plusieurs années de litige entre l'Université de Berkeley et AT&T. Les étudiants de l'université travaillent à remplacer les programmes développés par AT&T par leurs propres programmes, sous licence libre, afin de résoudre le litige. Cette situation dure jusqu'à la sortie de 4.4BSD en 1994, qui ne contient pratiquement plus de code AT&T.

En **1991**, Linus Torvalds, étudiant à l'université d'Helsinki, inspiré par les travaux de Tanenbaum, sort la toute première version (0.01) de son propre noyau : **Linux**, qui est au départ une **réécriture de Minix**. Linux passe sous licence GNU en 1992 et il faut attendre **1994** pour voir la **version 1.0**, donnant ainsi **naissance** à la **distribution** d'un **système d'exploitation entièrement libre**, **GNU/Linux**.

C'est à la suite des initiatives et travaux de Linus Torvalds et de Richard Stallman, aidés par des milliers de bénévoles, et consécutivement aux travaux des étudiants de l'université de Berkeley que **GNU/Linux et 4.4BSD** sont devenus les **premiers systèmes d'exploitation sous licence libre**.

IV. Rôles du système d’exploitation

Les services offerts par le système d’exploitation sont en rapport avec l'utilisation des ressources de l'ordinateur par les programmes. Ils permettent en particulier d'exécuter des programmes, de lire et écrire des informations, de manipuler les fichiers, de communiquer entre ordinateurs et de déceler des erreurs.

Ces services permettent à plusieurs usagers et plusieurs programmes de se partager les ressources de l'ordinateur. Le principal rôle du système d'exploitation est alors de **gommer les différences entre les différentes architectures informatiques**, et d'organiser l'utilisation des ressources de façon rationnelle.

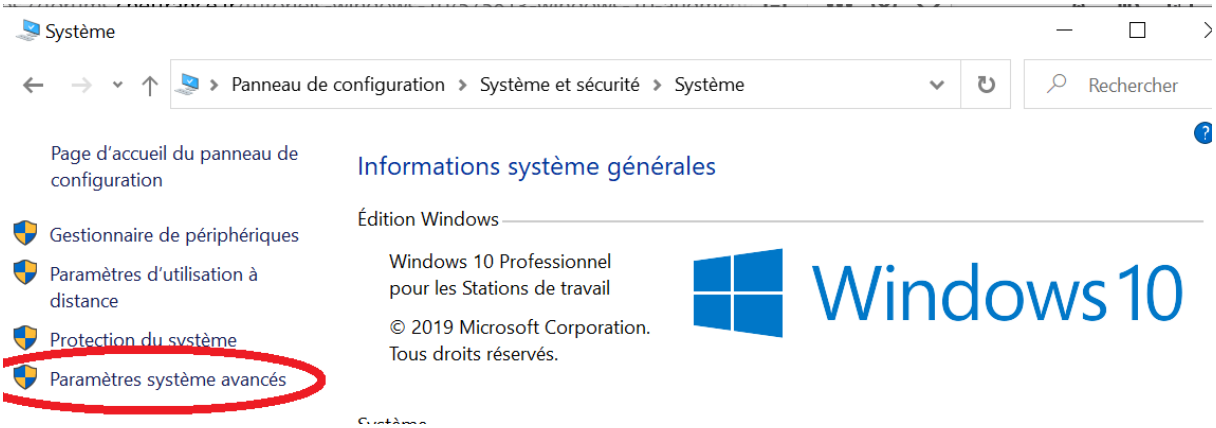
Les rôles du système d'exploitation sont divers :

- **Gestion des ressources.** Une des fonctions du système d'exploitation est de protéger les ressources contre l'utilisation par des personnes non autorisées, et de résoudre les conflits lorsque deux utilisateurs demandent simultanément la même ressource.

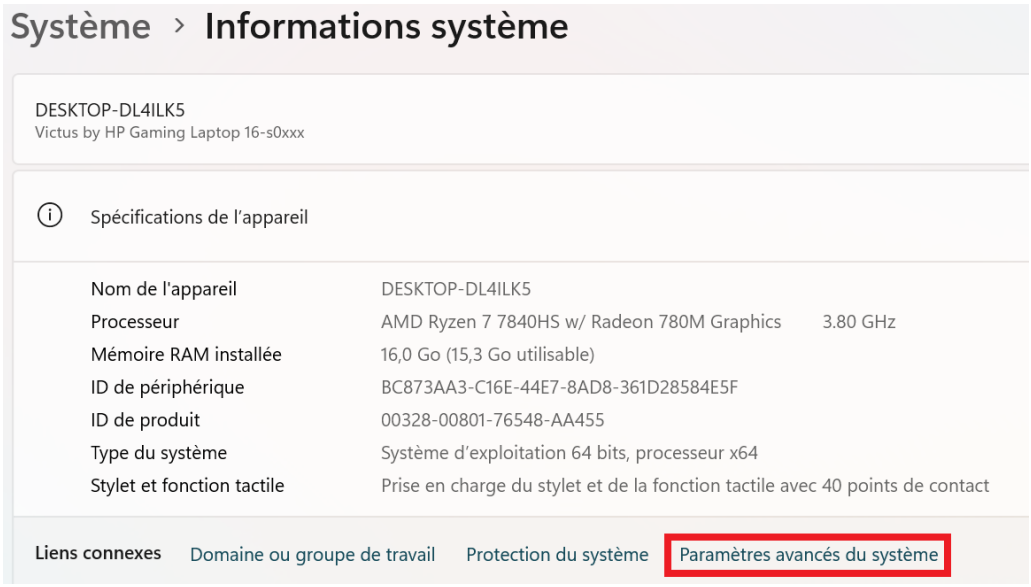
  - Gestion du processeur : le système d'exploitation est chargé de gérer l'allocation du processeur entre les différents programmes grâce à un **algorithme d'ordonnancement**. Le type d'ordonnanceur est totalement dépendant du système d'exploitation, en fonction de l'objectif visé.

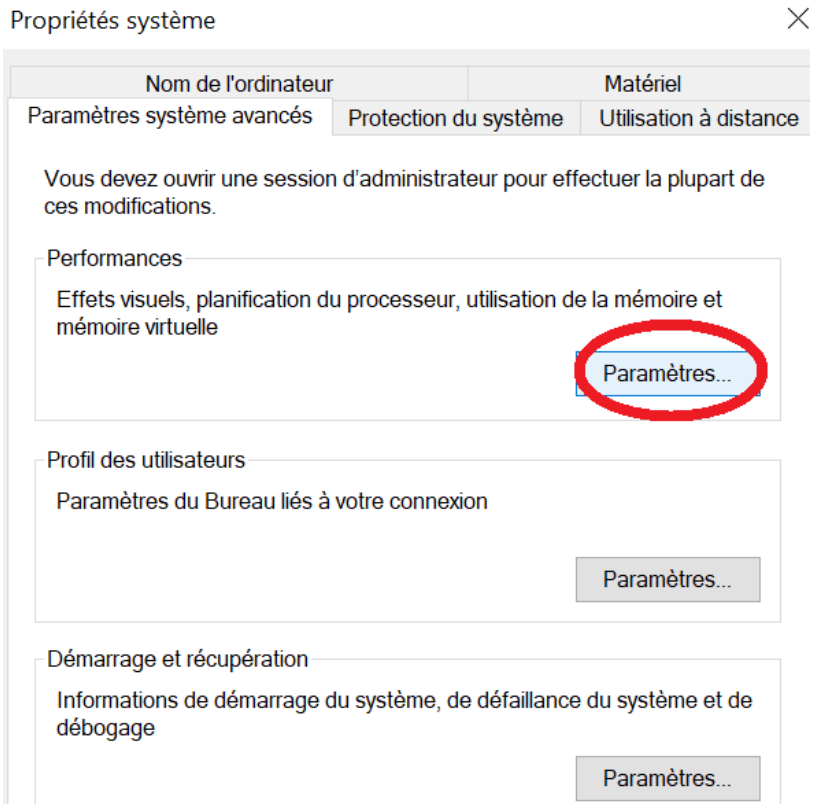
  - Gestion de la mémoire vive : le système d'exploitation est chargé de gérer l'espace mémoire alloué à chaque application et, le cas échéant, à chaque usager. En cas d'insuffisance de mémoire physique, le système d'exploitation peut créer une zone mémoire sur le disque dur, appelée **mémoire virtuelle**. La mémoire virtuelle permet de faire fonctionner des applications nécessitant plus de mémoire qu'il n'y a de mémoire vive disponible sur le système. En contrepartie cette mémoire est beaucoup plus lente. On parle aussi de **mémoire paginée (ou swapping)** qui permettra d'utiliser de la mémoire de masse comme extension de la mémoire vive ; augmenter le taux de multiprogrammation ; mettre en place des mécanismes de protection de la mémoire ; partager la mémoire entre processus.

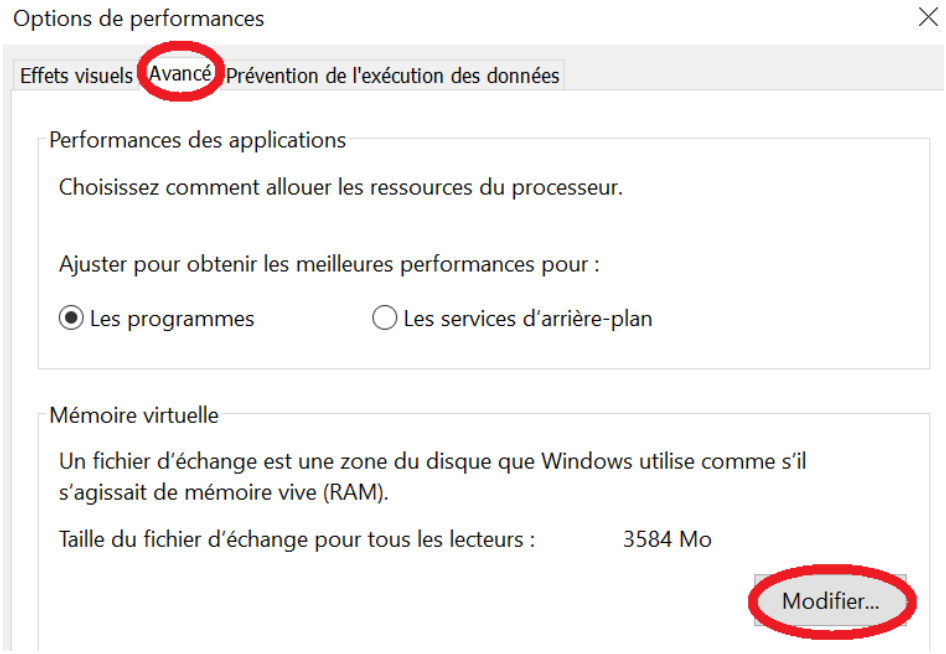
Manipulations :
Augmenter la mémoire paginée sur son ordinateur.
Panneau de configuration\Système et sécurité\Système :

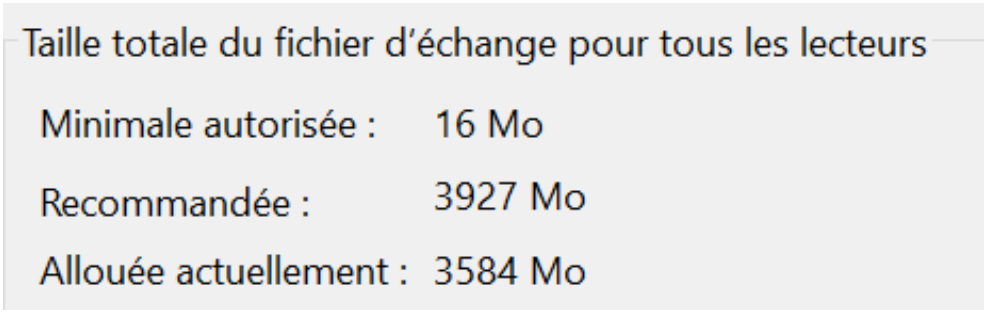


Sur Windows 11 :

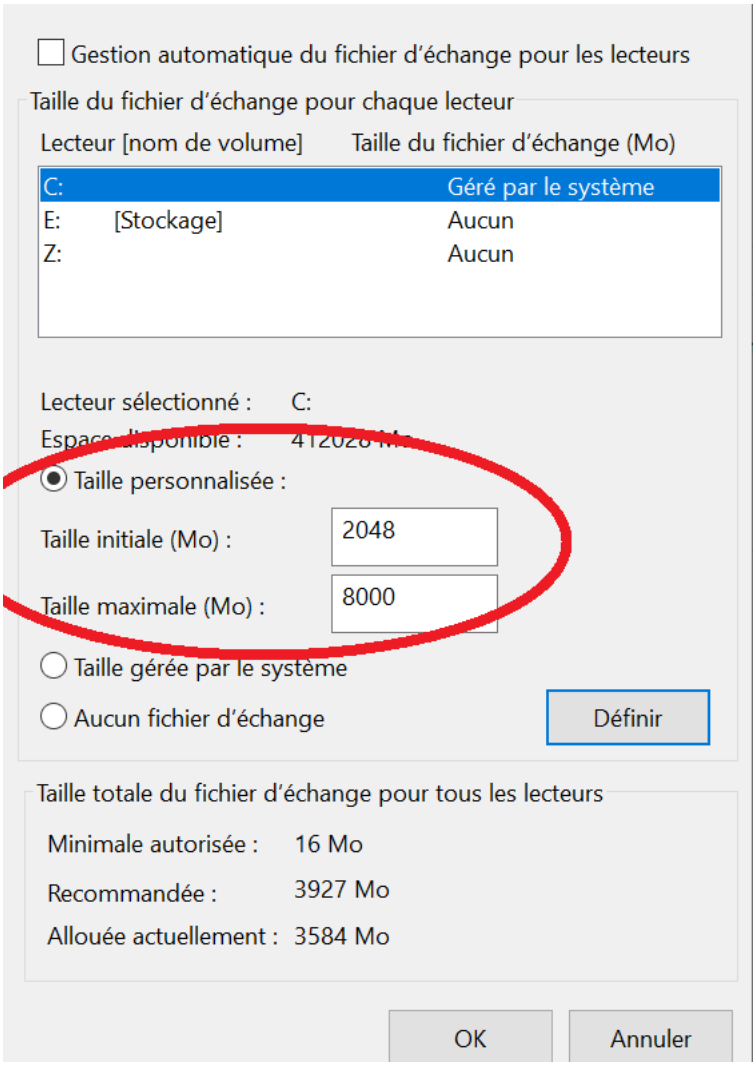








Plutôt que de laisser Windows gérer la « taille du fichier d'échange », on pourra le définir soit même en décochant « Gestion automatique du fichier d'échange pour les lecteurs ».
On pourra définir une « taille personnalisée » en rentrant par exemple la valeur recommandée indiquée pour la taille minimale et la taille maximale.



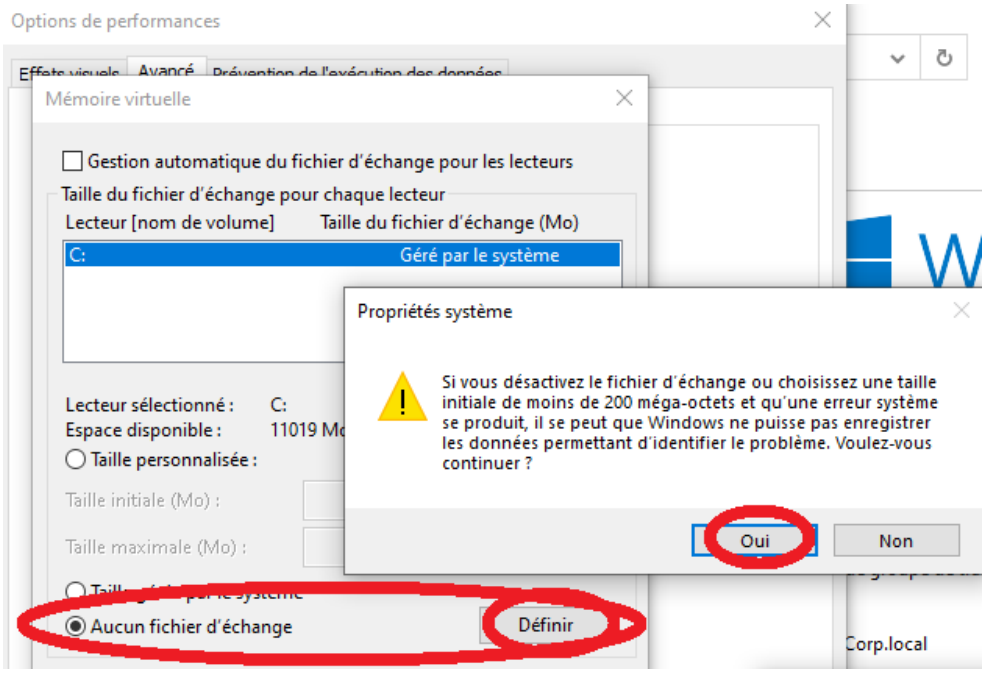
Dans l'exemple ci-dessus, on a diminué la taille minimale initiale du fichier d'échange, l'ordinateur étant équipé de 24Go de RAM. On peut en revanche augmenter un peu la taille maximale en cas de besoin (ici 8000Mo, soit jusqu'à 8Go d'espace disque dur utilisé par la mémoire virtuelle/paginée.

Installer VirtualBox (ou un virtualiseur que vous maîtrisez : VMWare, Hyper-V, Proxmox, etc) + Windows 10

VIRTUAL BOX SUR LE DISQUE DUR EXTERNE DANS LOGICIEL → SYSTEMES EXPLOITATION (récupérer **virtual box + extension : add-on invité**)
WINDOWS 10** : LOGICIEL → SYSTEMES DEXPLOITATION → WINDOWS →* ***Windows 10 (64 bits)

On pourra essayer de sous paramétrer cette mémoire pour faire planter notre Windows (attention a bien le faire sur la machine virtuelle et pas sur votre ordinateur hôte !!!)

Baisser la RAM attribuée à 1Go (VirtualBox → Configuration → Système) / 1,9Go ou 2Go si vous êtes sur Windows 11, puis supprimer la mémoire paginée (« aucun fichier d'échange » puis cliquez sur « Définir ») sur votre machine virtuelle une fois redémarrée.



Pour rétablir la machine virtuelle on va remettre 2Go de mémoire RAM au niveau de VirtualBox- puis attribuer de nouveau de l'espace pour le fichier d'échange (ou cocher la case pour laisser Windows gérer cela tout seul).

- **Gestion des entrées/sorties** : le système d'exploitation 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). **Utilisation des périphériques** : **chaque périphérique a ses propres instructions**, avec lesquelles il peut être manipulé. Le système d'exploitation en tient compte. Il permet au programmeur de manipuler le périphérique par de simples demandes de lecture ou d'écriture, lui évitant la perte de temps de traduire les opérations en instructions propres au périphérique.

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

- **Gestion des droits et accès aux ressources** : le système d'exploitation 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. Il doit aussi résoudre les conflits lorsque deux utilisateurs demandent simultanément la même ressource.

- **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. En plus des instructions propres à chaque périphérique (lecteur de disquette, disque dur, lecteur de CD-ROM / DVD), le système d'exploitation tient compte du format propre de chaque support servant au stockage de fichiers. Il offre également des mécanismes de protection permettant de contrôler quel utilisateur peut manipuler quel fichier.

Cf gestion des disques : partition NTFS et FAT si on branche clef USB ou disque dur externe

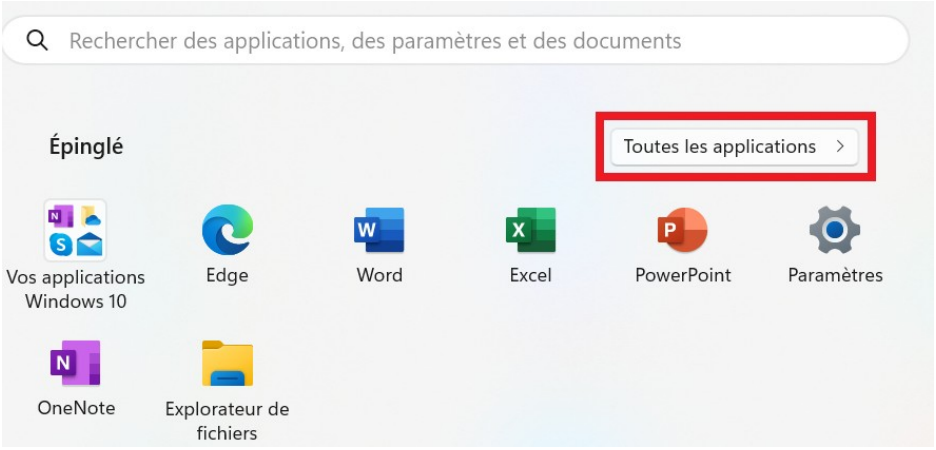
- **Contrôle - Gestion des informations** : le système d'exploitation fournit un certain nombre d'indicateurs permettant de diagnostiquer le bon fonctionnement de la machine. On a ainsi des statistiques d'utilisation des ressources, on peut surveiller les performances et les temps de réponse d'une machine cliente.

cf Gestionnaire des tâches → Performances de l'ordinateur

Clic droit sur la barre des tâches tout en bas puis gestionnaire des tâches ou « ctrl + maj + echap » ou encore « ctrl + maj + echap » pour ouvrir la fenêtre directement.

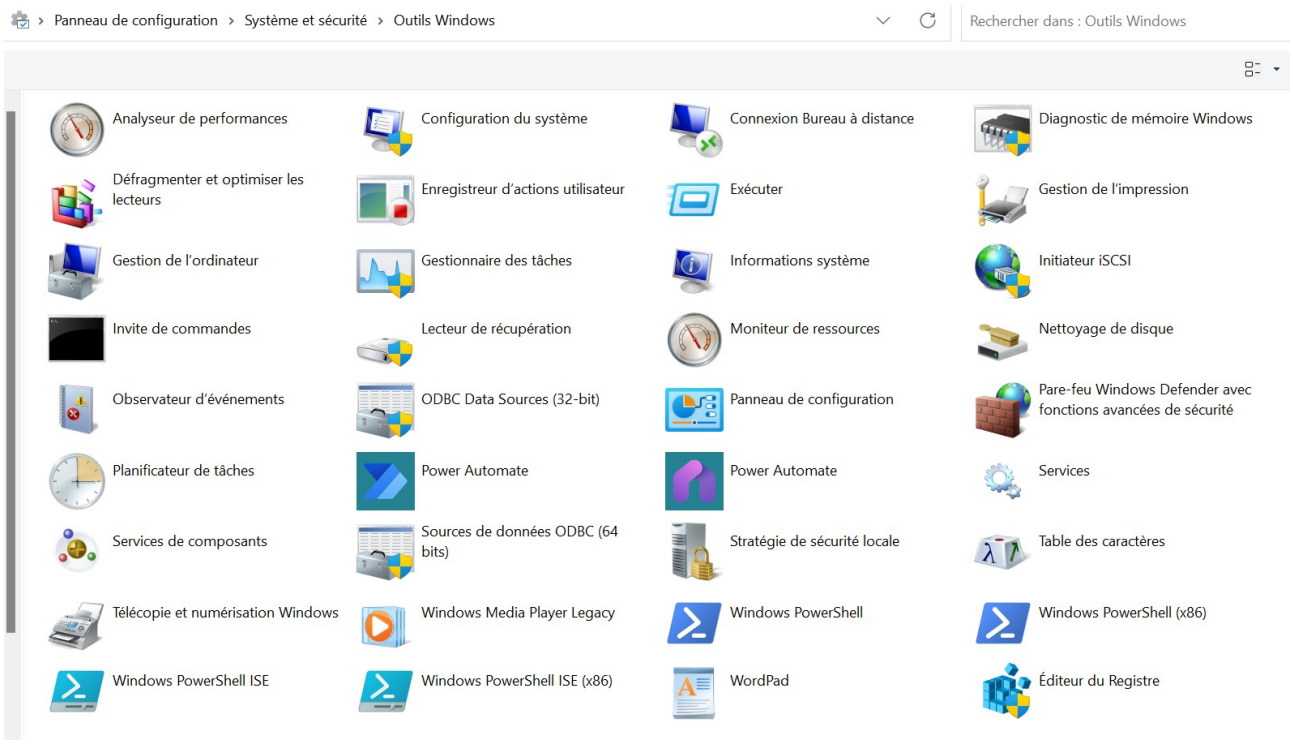
https://openclassrooms.com/fr/courses/5668856-exploitez-votre-pc-avec-windows-10/5669381-analysez-ce-qui-se-passe-sur-votre-pc

Attention, « Outils d'administration a changé en « Outils Windows » sur Windows 11. Touche « Windows » puis cliquez sur « Toutes les applications » :



Puis cherchez « Outils Windows » : 

Vous retrouvez cela via le panneau de configuration comme d'habitude :



- **Détection et récupération en cas d'erreur** : lorsqu'une erreur survient, qu'elle soit matérielle ou logicielle, le système d'exploitation traite l'erreur en adoucissant son impact sur le système informatique. Il peut tenter de réitérer l'opération, arrêter l'exécution du programme fautif, ou signaler le problème à l'utilisateur.

La palette des services offerts et la manière de s'en servir diffère d'un système d'exploitation à l'autre. Toutefois, le standard industriel **POSIX** du **IEEE** définit une suite d'appels systèmes standard. **POSIX** est une famille de **normes techniques** définie depuis **1988** par l'**Institute of Electrical and Electronics Engineers (IEEE)**. Ces normes ont émergé d'un projet de standardisation des interfaces de programmation des logiciels destinés à fonctionner sur les variantes du système d'exploitation UNIX.

Un logiciel applicatif qui effectue des appels système selon POSIX pourra être utilisé sur tous les systèmes d'exploitation conformes à ce standard.

Le système d'exploitation assure la réservation des différentes ressources pour les besoins des programmes exécutés simultanément. Les réservations peuvent être inscrites dans des journaux d'activités à des fins de statistiques ou de dépannage et le système d'exploitation peut refuser une réservation à un utilisateur n'ayant pas reçu d'autorisation préalable.

Le matériel informatique peut exécuter des **instructions**. La **traduction d'opérations complexes en suite d'instructions** est une tâche fastidieuse qui incombe au système d'exploitation. Le système

d'exploitation prend en charge toute la **manipulation du matériel**, le logiciel applicatif ne peut donc pas voir de différence entre une machine simple, rudimentaire et une machine riche et complexe : les mêmes services sont offerts dans les deux cas.

Le système d'exploitation facilite le travail de programmation en fournissant une suite de services pouvant être utilisés par les logiciels applicatifs. Du point de vue du programmeur, son logiciel applicatif s'oriente en direction du système d'exploitation et du matériel, et les programmes sont considérés comme fonctionnant sur le système d'exploitation. Un système d'exploitation peut ainsi être vu comme une machine virtuelle. L'ensemble composé du matériel et du système d'exploitation forme la « machine » qui exécute le logiciel applicatif, une machine en partie simulée par du logiciel.

Un système d'exploitation est composé de programmes diverses et variés. La composition exacte dépend de l'usage cible et du type d'appareil informatique auquel le système est destiné (ordinateur personnel, serveur, superordinateur/supercalculateur ou encore système embarqué).

A ce propos, les systèmes d'exploitation et programmes doivent être entièrement repensés et réécris pour pouvoir être utilisé sur les nouveaux **ordinateurs quantiques**.

Définition :

https://www.institut-pandore.com/physique-quantique/informatique-ordinateur-quantique/

Applications :

https://experiences.microsoft.fr/business/intelligence-artificielle-ia-business/ordinateur-quantique-en-2020/

Les rôles du système d'exploitation seront différents suivants la plateforme physique sur laquelle il se trouve.

Ainsi, nous savons que pour un ordinateur, il gère l'attribution et l'utilisation des ressources (processeurs et mémoire RAM par exemple). Il fait aussi fonctionner les périphériques (clavier, souris, surface tactile, écran, disque dur, lecteur de DVD, lecteur de cartes mémoire...).

Dans un appareil photo, il fera fonctionner les différents mécanismes et fonctionnalités : affichage de l'écran, actions de l'utilisateur, etc...

V. Composantes du système d’exploitation

Le système d'exploitation est composé d'un **ensemble de logiciels permettant de gérer les interactions avec le matériel**. Parmi cet ensemble de logiciels on distingue généralement les éléments suivants :

- Le **noyau** (**kernel**) représente les **fonctions fondamentales** du système d'exploitation telles que la gestion de la mémoire, des processus, des fichiers, des entrées-sorties principales, et des fonctionnalités de communication.
- L'**interpréteur de commande** (**shell**, «*coquille*» par opposition au noyau) permettant la **communication avec le système d'exploitation par l'intermédiaire d'un langage de commandes**, afin de permettre à l'utilisateur de piloter les périphériques en ignorant tout des caractéristiques du matériel qu'il utilise, de la gestion des adresses physiques, etc.

Cf Les principales commandes Unix : https://www.shellunix.com/commandes.html
Commandes DOS : ipconfig, ping, cd, dir, etc.

- Le **système de fichiers (file system, FS),** permettant d'enregistrer les fichiers dans une arborescence.

Un système d'exploitation est essentiellement événementiel (s’oppose à la notion de séquentiel) - il est exécuté lorsque quelque chose s'est passé, typiquement lors d'un appel système, une interruption matérielle ou une erreur. L’OS sera principalement défini par ses réactions aux différents événements qui peuvent se produire.

C'est un logiciel étendu et complexe, qui offre de nombreuses fonctions. Il est construit comme une suite de modules, chacun ayant une fonction déterminée.

Le **noyau (kernel)** est la **pièce centrale** du système d'exploitation. C'est le second programme chargé en mémoire (juste après le bootloader) et il y reste en permanence - ses services sont utilisés continuellement. Un **chargeur d'amorçage** (**bootloader**) est un logiciel permettant de lancer un ou plusieurs systèmes d'exploitation (multiboot), c'est-à-dire qu'il permet d'utiliser plusieurs systèmes, à des moments différents, sur la même machine.

Il réside généralement dans un **emplacement protégé de mémoire vive**, qui ne peut **pas être modifié ni exploité** par les autres programmes (c'est-à-dire dans le cas d'un système d'exploitation en mode protégé).

C'est un composant critique : si le kernel subit une erreur et s'arrête alors l'ordinateur cessera de fonctionner, tandis que si un autre programme s'arrêtait (par exemple un programme utilisateur) alors le système d'exploitation resterait opérationnel.

Il offre typiquement des fonctions pour **créer ou détruire des processus** (exécuter des programmes), diriger l'utilisation du processeur, de la mémoire et des périphériques. Il offre également les fonctions qui permettent aux programmes de communiquer entre eux et de s'aligner dans le temps (**synchronisation**).

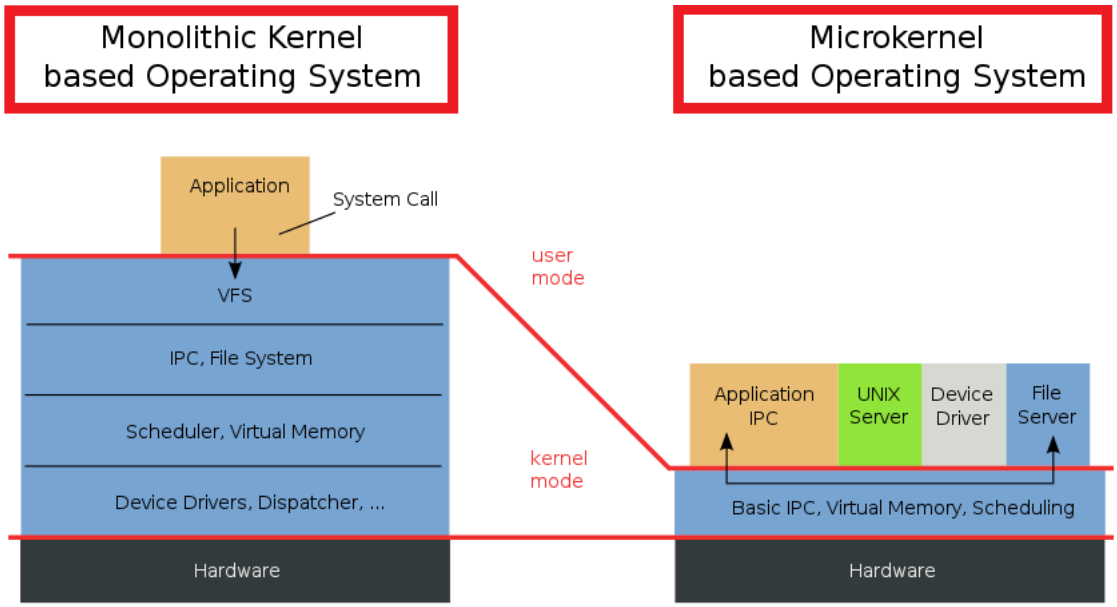
On peut distinguer différents types de noyaux :

- **Noyau monolithique** :

Dans une construction **monolithique**, le système d'exploitation est composé d'**un seul programme** : le noyau. Celui-ci est typiquement organisé en couches. La construction monolithique est la plus courante, appliquée à la plupart des systèmes Unix.

- **Noyau à micronoyau :**

Dans la construction **micronoyau**, le kernel fournit les services minimum : de nombreuses fonctions du système d'exploitation ont été retirées du noyau et sont offertes par des programmes manipulés par celui-ci, qui sont ce que l'on appelle des **services** (pour un système en mode protégé, la grande différence avec un système monolithique sera que ces services seront exécutés dans l'espace mémoire utilisateur et non celui du noyau).
Les **appels de fonction** au système d'exploitation par les programmes utilisateurs ont été **remplacées** par des **envois de messages**. Dans cette construction le noyau est utilisé principalement pour planifier l'exécution de processus, et pour échanger des messages. AIX, BeOS, Mach, Hurd, MacOS X, Minix et QNX sont des systèmes d'exploitation qui utilisent cette construction par exemple.



- **Noyau hybride :**

La construction **hybride** ressemble à une construction **micronoyau (microkernel)**, cependant certaines fonctions ont été placées dans le noyau pour des raisons d'efficacité. Windows NT (premier « WindowsServer », 2000, XP et les Windows plus récents (notamment Windows 10 et Windows-Server2019) ainsi que DragonFly BSD sont en construction hybride;

- Système organisé par couches :

Le principe de la **répartition par couches** est que **chaque module d'une couche donnée utilise uniquement des fonctions offertes par les modules qui se trouvent dans la couche au-dessous**. L'organisation en couche n'est pas unique aux systèmes d'exploitation et est **couramment appliquée** aux **logiciels applicatifs**. Dans un système d'exploitation, les couche **inférieures** concernent les **périphériques** et la **mémoire** ; au-dessus desquelles se trouvent les **systèmes** de **fichiers**, puis les **processus**. Ces couches constituent la normalisation de la machine : pour un programme utilisateur et un système d'exploitation donné, tous les ordinateurs seront identiques (ou presque). C'est ce que l'on nomme l'**abstraction matérielle.**

- Appels système :

Le **kernel étant dans un emplacement protégé**, il est impossible pour un logiciel applicatif d'appeler directement ses fonctions. Un mécanisme permet aux logiciels applicatifs de demander des services au système d'exploitation. Il est typiquement mis en œuvre par une bibliothèque. Celle-ci comporte des **fonctions bouchon** qui consistent à placer les paramètres selon une convention, puis utiliser une instruction du processeur qui provoque la mise en pause du processus en cours et l'exécution du système d'exploitation. Les fonctions de bouchon s'utilisent comme les fonctions ordinaires d'une bibliothèque.

- Noyau client-serveur :

Dans une construction **client-serveur**, le cœur du système d'exploitation a pour seule fonction d'**assurer la communication entre les modules**. Le système d'exploitation est divisé en nombreux petits modules qui sont exécutés de la même manière que des logiciels applicatifs. Cette construction est bien adaptée aux systèmes d'exploitation distribués;

Un **système d'exploitation distribué** (notion de **répartition**) est une couche logicielle au dessus d'un ensemble de **nœuds de calculs indépendants** (ordinateurs, serveurs, supercalculateurs, etc...), **communiquant** par un système de **réseau** propre ou général.

- Principe de machine virtuelle :

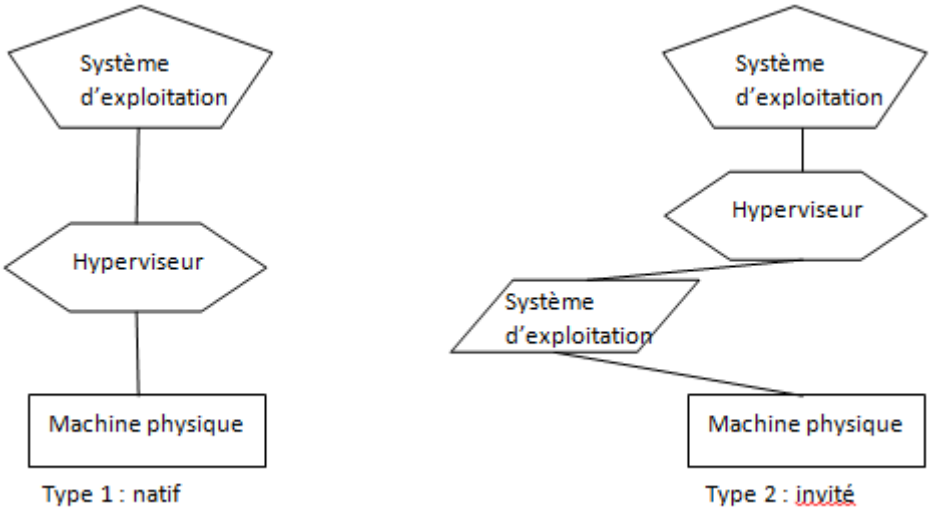
Le principe des **machines virtuelles**, est que le système d'exploitation crée l'illusion qu'il existe plusieurs machines, aux capacités étendues, en utilisant une seule machine aux capacités plus limitées. Le cœur du système d'exploitation est un moniteur qui crée les illusions — machine virtuelle. Les caractéristiques techniques de ces machines virtuelles sont identiques à celle de la machine réellement utilisée et celles-ci peuvent être utilisées pour exécuter un autre système d'exploitation. On parle aussi d'**hyperviseurs**.

Un **hyperviseur** est une plate-forme (logiciel) de virtualisation qui permet à plusieurs systèmes d'exploitation de travailler sur une même machine physique en même temps en créant des machine virtuelles (virtual machines : Vms).

Il existe 2 types d'hyperviseurs :

- Le type 1 (natif, bare metal)

- Le type 2 (invité, hébergé)



Certains hyperviseurs comme Hyper-V peuvent être des 2 types. Dans notre cas, VirtualBox est installé sur notre système d'exploitation donc c'est bien un hyperviseur de type 2.

VI. Typologie – type de systèmes d’exploitation

On peut distinguer cinq générations de systèmes d'exploitation : *par* **lots** *(***batch***)*, **multi programmés**, en **temps partagé**, **temps réel**, et **distribués**. Chacun des principes mis en œuvre dans une génération se retrouve dans les générations suivantes et nous allons les décrire ci-dessous dans l’ordre chronologique.

Un système de traitement par **lots** (***batch***) est prévu pour **l'exécution** de **grands calculs** les uns après les autres, avec peu d'intervention utilisateur.

Ces systèmes d’exploitation sont apparus dans les années **1950**. Un programme et ses données n'est rien d'autre qu'une pile de cartes avec des indicateurs de début et de fin de lot. L'exécution d'un programme consiste à demander à un opérateur de placer la pile de cartes dans le lecteur, puis l'opérateur lance la lecture séquentielle des cartes. Le processeur central est au repos, durant les manipulations de l'opérateur.

Un **batch** est un **lot de travaux** à effectuer. L'opérateur compose un batch en posant les unes sur les autres les piles de cartes des différents programmes et leurs données demandés par les utilisateurs. Il forme une grande pile de cartes séparées par des marque-page, en général une carte de couleur particulière, qu'il place ensuite dans le lecteur. Le regroupement de plusieurs programmes en un batch diminue les interventions de l'opérateur.

Dans un système basé sur les *batchs*, le cœur du système d'exploitation est un **programme moniteur qui réside continuellement en mémoire centrale** et permet à l'opérateur de demander le début ou l'arrêt de l'exécution du lot. À la fin de l'exécution de chaque tâche du lot, le moniteur effectue des travaux de nettoyage, puis lance l'exécution de la tâche suivante. Ainsi, l'opérateur intervient uniquement au début et à la fin du lot.

Dans ces systèmes d'exploitation les commandes ajoutées au marque-page, formulées dans le langage **JCL (*Job Control Language*)** sont un des seuls moyens qu'a l'utilisateur d'interagir avec le système d'exploitation.

Les systèmes d'exploitation *batch* sont adaptés à des applications nécessitant de **très gros calculs mais peu d'implication de l'utilisateur** : météo, statistiques, impôts, simulations en général... Les utilisateurs n'attendent pas immédiatement de résultats. Ils soumettent les demandes, puis reviennent ultérieurement collecter les résultats.

En raison de la grande différence de vitesse entre le processeur et les périphériques, dans un système d'exploitation **batch** le **processeur est inutilisé 90 % du temps** car les programmes **attendent** qu'un périphérique ou un autre termine les opérations. Avec ces systèmes d'exploitation il n'y a pas de concurrence entre les différentes tâches, la mise en œuvre de l'utilisation du processeur, de la mémoire et des périphériques est triviale mais loin d’être optimale.

À partir de la génération des systèmes d'exploitation **multitâches** ou **multi-programmés**, **plusieurs programmes sont exécutés simultanément par planification (scheduling)**. Dans ces systèmes d'exploitation multitâches, plusieurs programmes résident dans la mémoire centrale et le système d'exploitation suspend régulièrement l'exécution d'un programme pour continuer l'exécution d'un autre.

Dans la génération des systèmes **multi-programmés**, l'exécution simultanée de plusieurs programmes vise l'utilisation efficace de la puissance de calcul du processeur.

Ces systèmes sont apparus dans les années **1960**. Le but recherché par de tels systèmes est d'augmenter l'efficacité de l'utilisation du processeur et des périphériques en utilisant la possibilité de les **faire fonctionner en parallèle**. Plusieurs programmes sont placés en mémoire centrale, et lorsque le programme en cours d'exécution attend un résultat de la part d'un périphérique, le système d'exploitation ordonne au processeur d'exécuter un autre programme.

Dans les systèmes d'exploitation multi-programmés, l'utilisation du processeur est partagée par **planification (scheduling)** : à chaque utilisation d'un périphérique, le système d'exploitation choisit quel programme va être exécuté. Ce choix se fait sur la base de priorités. Le système d'exploitation comporte un **mécanisme de protection** évitant ainsi que le programme en cours d'exécution ne lise ou n'écrive dans la mémoire attribuée à un autre programme. Les programmes sont exécutés dans un mode **non-privilégié**, dans lequel l'exécution de certaines instructions est interdite.

Les systèmes multitâches nécessitent un ordinateur et des périphériques mettant en œuvre la technique du **DMA (direct memory access)** celle-ci, le processeur ordonne à un périphérique d'effectuer une opération, le résultat de l'opération est placé en mémoire centrale par le périphérique tandis que le processeur exécute d'autres instructions. Dans les systèmes multi-programmés, tout comme pour les systèmes batch, l'utilisateur n'a que peu de contact avec les programmes et peu de possibilités d'intervention.

Dans la génération des systèmes en **temps partagé** l'exécution simultanée de plusieurs programmes vise à **répondre rapidement aux demandes de plusieurs utilisateurs** en communication directe avec l'ordinateur. On a **l'illusion** que toutes les tâches sont exécutées en même temps.

Les systèmes d'exploitation en temps partagé sont apparus dans les années **1970**. Ils sont utilisés dans des dispositifs interactifs où plusieurs utilisateurs sont simultanément en dialogue avec l'ordinateur. Un système d'exploitation en **temps partagé** est destiné à r**épondre rapidement aux demandes de l'utilisateur**, et donner à chaque utilisateur l'impression qu'il est le seul à utiliser l'ordinateur.

Un système en temps partagé met en œuvre des techniques sophistiquées de **multiprogrammation** en vue de permettre l'utilisation interactive de l'ordinateur par plusieurs utilisateurs et plusieurs programmes simultanément.

Dans ces systèmes, tout comme dans la génération précédente, l'**utilisation du processeur est planifiée**. Cependant, contrairement aux systèmes de la génération précédente, dans les systèmes en temps partagé **chaque programme est exécuté durant une tranche de temps déterminé,** puis le système d'exploitation bascule sur l'exécution d'un autre programme, ce qui évite qu'un programme monopolise l'utilisation du processeur au service d'un utilisateur, entraînant des retards pour les autres utilisateurs.

Les systèmes d'exploitation en temps partagé mettent en œuvre la technique du **swap** : lorsque le programme en cours d'exécution a besoin de plus de mémoire que celle disponible, un autre programme inactif est retiré pour gagner de la place, le programme inactif est alors enregistré temporairement sur le disque dur. L'enregistrement sur disque provoque cependant une perte de temps non négligeable.

De nombreux systèmes d'exploitation sont basés sur **Unix**, un système en temps partagé.

Un système d'exploitation **temps réel (real time systems**) doit garantir que toute opération se termine dans un **délai donné**, en vue de garantir la réussite du dispositif dans lequel l'ordinateur est utilisé.

Il sont essentiellement utilisés dans l'industrie. Un système temps réel doit 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). Ces systèmes d'exploitation sont souvent utilisés par des **ordinateurs reliés à un appareil externe** (pilotes automatiques, robots industriels, applications vidéo et audio) pour lequel un retard de réponse de l'ordinateur entraînerait un échec de l'appareil.

Ces systèmes sont apparus au **milieu des années 1970**.

Certains services offerts par ces systèmes d'exploitation sont réalisés comme des logiciels applicatifs, et sont exécutés en concurrence avec ceux-ci. Un système d'exploitation temps réel autorise un **contact direct** entre les logiciels applicatifs et les périphériques. Dans certains systèmes temps réel les ressources sont réservées, évitant ainsi les ralentissements que provoqueraient les réservations à la volée, et garantissant que les ressources sont continuellement disponibles.

Les systèmes d'exploitation temps-réel évitent d'utiliser la technique du swap en raison des risques de dépassement des délais.

Un **système distribué dirige l'utilisation des ressources de plusieurs ordinateurs à la fois**. Il utilise les capacités d'un réseau informatique, contrôle un groupe de machines, et les fait apparaître comme une machine unique, virtuelle, de très grande capacité.

La baisse des prix du matériel informatique a permis, dans les années **1990**, la création de **systèmes informatiques composés de plusieurs ordinateurs**, et donc plusieurs processeurs, plusieurs mémoires, et de nombreux périphériques. Un système distribué permet le partage des ressources entre les ordinateurs. Un utilisateur d'un ordinateur bon marché peut se servir de ressources coûteuses existant sur un autre ordinateur.

Chaque système d'exploitation est conçu pour fonctionner avec une gamme particulière de machines (type de processeur, constructeur, architecture). Si un système d'exploitation est disponible pour plusieurs gammes de machines différentes, alors le même code source est compilé et adapté à chaque gamme de machines. La palette de pilotes inclus dans le système d'exploitation est adaptée au matériel informatique disponible sur le marché pour cette gamme de machines.

VII. Fonctionnalités dans la pratique

Nous avons vu que les utilisateurs et les programmeurs pouvaient demander des services au système d'exploitation par son **interface de programmation** (les appels système permettent des interactions entre un programme en cours d'exécution et le système d'exploitation. L'utilisation d'appels système ressemble a celle de fonctions), ses **commandes** ou son **interface graphique**.

1 Processeur

Nous avons vu qu'un système d'exploitation **multitâches** permet à plusieurs utilisateurs de se servir de l'ordinateur et donne à chaque utilisateur l'impression qu'il est le seul à utiliser l'ordinateur. Pour ce faire, l'**utilisation du processeur est planifiée** : chaque programme est exécuté durant une tranche de temps déterminé, puis le système d'exploitation bascule sur l'exécution d'un autre programme.

Un **processus** est un **ensemble d'instructions qui sont en train d'être exécutées**. Les instructions proviennent d'un programme, et l'exécution nécessite du temps, de la mémoire, des fichiers et des périphériques. Le système d'exploitation s'occupe de **créer, d'interrompre, et de supprimer des processus**. Plusieurs processus se trouvent en mémoire centrale en même temps.

La responsabilité du système d'exploitation et de **réserver** de la **mémoire**, et de **planifier l'exécution**, de **s'occuper** des **interblocages** et **d'assurer** les **communications** entre les processus. **L'ordonnanceur (scheduler)** associe un processus à un processeur, puis plus tard le dissocie du processeur pour associer un autre processus. cette opération associer / dissocier est appelée **context switch**. Lors de la **planification**, le système d'exploitation tient compte de la **disponibilité**, ou non, des ressources utilisées par le processus. Certains systèmes d'exploitation créent des processus pour effectuer certaines tâches propre au système.

L'**ordonnanceur (scheduler)** désigne le composant du noyau du système d'exploitation choisissant l'ordre d'exécution des processus sur les processeurs d'un ordinateur.

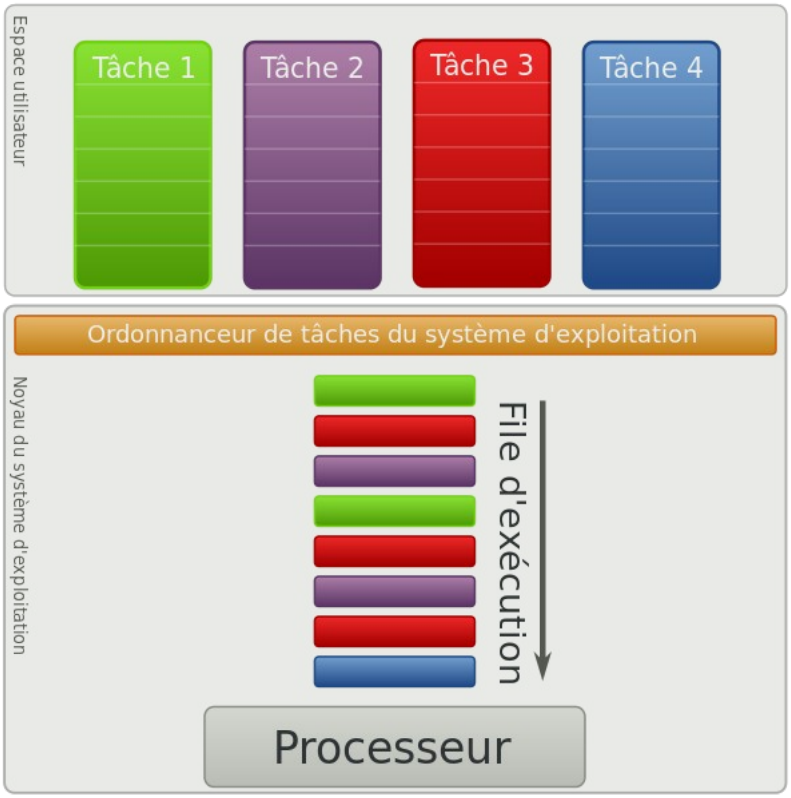
Un processus a besoin de la ressource processeur pour exécuter des calculs; il l'abandonne quand se produit une interruption, etc. De nombreux anciens processeurs ne peuvent effectuer qu'un traitement à la fois. Pour les autres, un ordonnanceur reste nécessaire pour **déterminer quel processus sera exécuté sur quel processeur** (c'est la notion d'affinité, très importante pour ne pas dégrader les performances). Au-delà des classiques **processeurs multicœur**, la notion **d'hyperthreading** rend la question de l'ordonnancement encore un peu plus complexe.

L'Hyper-Threading (**Hyper-Threading Technology** (**HTT**) ou **HT Technology** (**HT**)) est la mise en œuvre par l'entreprise Intel du simultaneous **multithreading** (SMT) à deux voies dans ses microprocesseurs. Il consiste à créer **deux processeurs logiques sur une seule puce**, chacun doté de ses **propres registres** de **données** et de **contrôle**, et d'un contrôleur d'interruptions particulier. Ces deux unités partagent les éléments du cœur de processeur, le cache et le bus système. Ainsi, **deux sous-processus peuvent être traités simultanément par le même processeur**. Cette technique multitâche permet d'utiliser au mieux les ressources du processeur en garantissant que des données lui soient envoyées en masse. Elle permet aussi d'**améliorer les performances** en cas de **défauts de cache (cache misses)**.

À un instant donné, il y a souvent davantage de processus à exécuter que de processeurs.

Un des rôles du système d'exploitation, et plus précisément de **l'ordonnanceur** du noyau, est de permettre à tous ces processus de s'exécuter à un moment ou un autre et d'utiliser au mieux le processeur pour l'utilisateur. Pour que chaque tâche s'exécute sans se préoccuper des autres et/ou aussi pour exécuter les tâches selon les contraintes imposées au système (exemple: contraintes temporelles dans le cas d'un système d'exploitation temps réel) , l'ordonnanceur du noyau du système effectue des **commutations de contexte** de celui-ci.

L'ordonnancement peut être de type **coopératif** : les tâches doivent être écrites de manière à **coopérer les unes avec les autres** et ainsi accepter leur suspension pour l'exécution d'une autre tâche. L'ordonnancement peut être également de type **préemptif** : l'ordonnanceur a la responsabilité de l'interruption des tâches et du choix de la prochaine à exécuter. Certains noyaux sont eux-mêmes préemptifs : l'ordonnanceur peut interrompre le noyau lui-même pour faire place à une activité (typiquement, toujours dans le noyau) de priorité plus élevée.



Nous avons vu que l’ordonnancement serait différent dans des systèmes de type temps partagé ou temps réel à cause des contraintes de temps du système temps réel. Les algorithmes utilisés pour l’ordonnancement seront donc différent.

Dans certains logiciels applicatifs, **plusieurs programmes effectuent la même tâche simultanément, et s'échangent des informations**.

La **communication inter-processus (inter-process communication, IPC)** regroupe un ensemble de mécanismes permettant à des **processus concurrents de communiquer**. Ces mécanismes peuvent être classés en trois catégories :

- les mécanismes permettant l'**échange de données** entre les processus ;
- les mécanismes permettant la **synchronisation entre les processus** (notamment pour gérer le principe de **section critique**) ;
- les mécanismes permettant l'échange de données *et* la synchronisation entre les processus.

Les **fichiers** peuvent être utilisés pour **échanger des données** entre plusieurs processus concurrents.

Un **fichier informatique** est une collection, un ensemble de données numériques réunies sous un même nom, enregistrées sur un support de s**tockage permanent**, appelé **mémoire de masse**, tel qu'un disque dur, un dvd, une mémoire flash ou une bande magnétique, et manipulées comme une unité. C’est une suite d'**informations binaires**, c'est-à-dire une suite de 0 et de 1. Ce fichier peut être stocké pour garder une trace de ces informations. Un fichier texte est un fichier composé de caractères stockés sous la forme d'octets. Un **octet** est un multiplet (byte) de 8 bits codant une information.

En vue de faciliter leur organisation, les fichiers sont disposés dans des systèmes de fichiers(NTFS, FAT, etc...) qui permettent de placer les fichiers dans des emplacements appelés répertoires ou dossiers eux-mêmes organisés selon le même principe de manière à former une **hiérarchie arborescente**.

Les processus voulant envoyer des données écrivent dans un ou plusieurs fichiers à certaines positions ; les processus souhaitant recevoir ces données se positionnent à ces positions dans le ou les fichiers et les lisent. Ce type d'échange est possible entre des **processus concurrents locaux en utilisant le système de fichiers local**, ou des **processus concurrents distants** en utilisant un système de **fichiers distribué**, tel que **NFS**.

La **mémoire principale** d'un ordinateur peut aussi être utilisée pour échanger des données entre plusieurs processus concurrents. Suivant le type de processus, les mécanismes utilisés ne sont pas les mêmes :

- dans le cas de **processus lourds (process**), les **espaces mémoires des processus ne sont pas partagés**. On utilise alors un mécanisme de partage de mémoire, tel que les segments de mémoire partagée dans Unix ;
- dans le cas de **processus légers (thread*)*, l'**espace mémoire des processus est partagé**, la mémoire peut donc être utilisée directement.

Dans les deux cas, les échanges sont réalisés en plaçant les données en mémoire dans des variables partagées par les processus.

Quelle que soit la méthode utilisée pour échanger les données (fichiers ou mémoire principale), ce type de communication pose le problème des **sections critiques** : le moment où les processus accèdent aux données partagées. En effet, si deux processus accèdent en même temps à une donnée commune, il peut se produire différents résultats :

- les d**onnées ne sont plus cohérentes** ;
- un ou plusieurs des **processus** concernés **plantent** ;
- un ou plusieurs des **processus** est **interrompu** : il doit attendre que la donnée commune soit libérée.

En utilisant des fichiers, on tombe généralement sur le deuxième ou le troisième cas. Si on le prévoit, le processus peut attendre (10 millisecondes, 1 seconde, etc.) et reprendre plus tard l'accès aux données. Cela dit, cette solution n'est pas toujours possible en réseau, car les fichiers ne sont pas toujours libérés correctement.

En utilisant la mémoire principale, on tombe plutôt sur le premier cas. Si on le prévoit, le processus peut effectuer des synchronisations par lectures/écritures exclusives. Dans tous les cas, le partage de données en mémoire n'est possible que sur un seul et même ordinateur.

Les mécanismes de **synchronisation** sont utilisés pour r**ésoudre les problèmes de sections critiques** et plus généralement pour **bloquer** et **débloquer** des **processus** suivant certaines conditions.

Les **verrous** permettent de **bloquer tout ou une partie d'un fichier**. Ces blocages peuvent être réalisés soit pour les opérations de lecture, soit d'écriture, soit pour les deux.

Les **sémaphores** sont un mécanisme plus général, ils ne sont pas associés à un type particulier de ressource et permettent de limiter l'accès concurrent à une section critique à un certain nombre de processus. Pour ce faire les sémaphores utilisent deux fonctions : P et V, et un compteur. La fonction P décrémente le compteur, si le compteur est nul le processus est bloqué. La fonction V incrémente le compteur et débloque l'un des processus bloqués.

Les **signaux** sont à l'origine destinés à **tuer** (**terminer**) un **processus** dans certaines conditions, par exemple le signal SIGSEGV tue un processus qui effectue un accès à une zone de mémoire qu'il n'a

pas allouée. Les signaux peuvent cependant être déroutés vers d'autres fonctions. Le blocage d'un processus se fait alors en demandant l'attente de l'arrivée d'un signal et le déblocage consiste à envoyer un message au processus.

Le problème des mécanismes de synchronisation est que les processus ne sont bloqués que s'ils les utilisent. De plus, leur utilisation est difficile et entraîne des **problèmes d'interblocage (tous les processus sont bloqués)**.

Il existe des situations usuelles de synchronisation lors de coopération inter-processus :

- **exclusion mutuelle** : ressource accessible par **un seul processus** à la fois. Exemple : carte bancaire, carte son.
- **cohorte** : ressource accessible par **N processus** à la fois. Exemple : parking pouvant accueillir 500 voitures.
- **rendez-vous** : ressource accessible après l'**attente de plusieurs processus**. Exemple : processus devant échanger des informations entre les étapes de l'algorithme.
- **producteurs–consommateurs** : ressource accessible **après la fin d'un autre processus.** Exemple : Formule 1 qui ne repart que lorsque les mécaniciens ont terminé, réception de données sur le réseau puis traitement.
- **lecteurs–rédacteurs** : ressource accessible par **une seule catégorie de processus** à la fois. Exemple : fichier pouvant être lu par plusieurs personnes si personne ne le modifie.

Il existe pour finir des mécanismes qui regroupent les possibilités des deux catégories précédentes et sont plus simples d'utilisation.

L'idée de ce type de mécanisme est de communiquer en utilisant le principe des **files (piles)**, les processus voulant envoyer des informations les placent dans la file ; ceux voulant les recevoir les récupèrent dans cette même file. Les opérations d'écriture et de lecture dans la file sont bloquantes et permettent donc la synchronisation.

Ce principe est utilisé par les files d'attente de message (message queue) sous Unix, par les sockets Unix ou Internet, par les tubes, nommés ou non, et par la transmission de messages (message passing).

Le mécanisme de protection de la mémoire (que nous allons détailler juste après) **empêche les programmes de manipuler les mêmes informations**, et ceux-ci doivent faire appel à des services du système d'exploitation.

Par mesure de sécurité, le système d'exploitation **réserve à chaque programme un espace d'adressage.** C’est un emplacement en mémoire que seul le programme en question peut manipuler. Le système d'exploitation détecte toute tentative d'accès en dehors de l'espace d'adressage et provoque l'arrêt immédiat du programme qui tente d'effectuer de telles opérations. On nomme cela une **erreur de protection générale**.

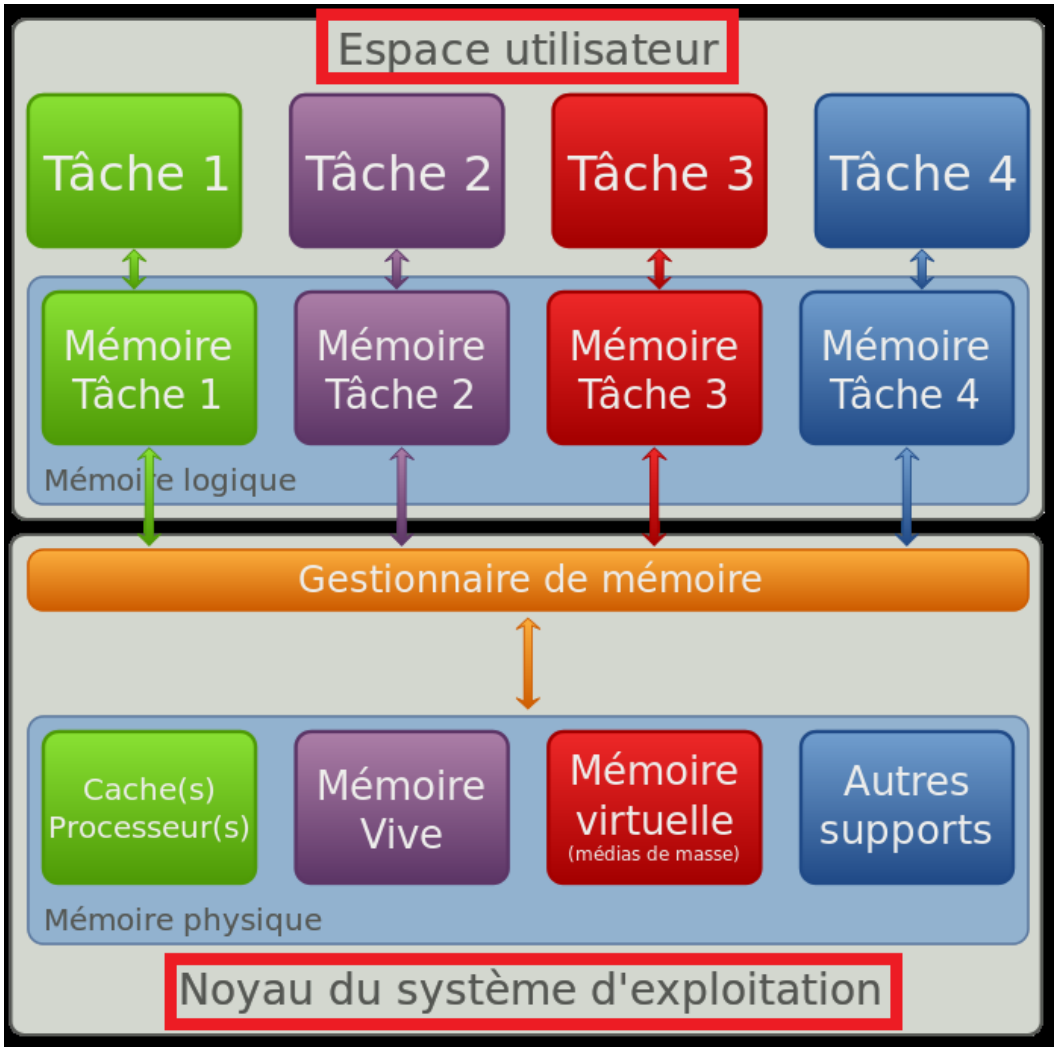
Observez les processus sur votre machine virtuelle et « tuez en » (fin de tâche) jusqu’à planter votre machine virtuelle.

2 Mémoire

La mémoire physique sur un système se divise en plusieurs catégories :

- la **mémoire vive** : composée de circuit intégrés, donc très rapide
- la **mémoire de masse** : composée de supports magnétiques (disque dur, bandes magnétiques...), beaucoup plus lente
- **Cache processeur**
- **Mémoire virtuelle** sur la mémoire de masse (swap, mémoire paginée).

La mémoire physique sert de zone de stockage temporaire pour les programmes et données que vous utilisez. De façon générale, plus la quantité de mémoire est importante, plus vous pouvez lancer d'applications simultanément. D'autre part, plus celle-ci est rapide plus votre système réagit vite, il s'agit donc pour le système d'exploitation de l'organiser au mieux pour en tirer le maximum de performances.



Le **gestionnaire de mémoire** est le sous-ensemble du système d'exploitation qui permet de gérer la mémoire de l'ordinateur. Sa tâche la plus basique est d'**allouer de la mémoire à des processus** lorsqu'ils en ont besoin. Cette mémoire allouée est par défaut propre au processus qui en fait la demande.

Sur les noyaux récents, le gestionnaire de mémoire va masquer la localisation physique de la mémoire (mémoire vive ou disque dur, dans l'espace de mémoire paginée) et présente au programme une **mémoire globale uniforme dite mémoire virtuelle**. Ainsi, tout processus croit manipuler une **mémoire logique** qui a les propriétés suivantes :

- la mémoire peut être étendue jusqu'aux capacités théoriques de la machine;
- la mémoire est privée (protégée), un processus ne peut pas accéder à la mémoire d'un autre processus (sauf allocations et autorisations spécifiques).

L'intérêt de ne pas indiquer au processus l'emplacement physique des données est de permettre au gestionnaire de mémoire de **placer et déplacer à sa convenance les données en mémoire, sans affecter les processus**. Ces données peuvent notamment être **fragmentées** dans la mémoire vive lorsqu'un processus demande un bloc de mémoire d'une taille supérieure au plus grand bloc physique libre. Le contenu de la mémoire peut aussi être migré. Cette **migration** est faite sur les différents supports mémoires tels que dans la mémoire physique (plus ou moins proche du processeur), dans la mémoire paginée, dans la mémoire accessible par réseaux (grappe de calcul).

La virtualisation de la mémoire permet aussi une gestion optimiste des ressources : la mémoire allouée mais pas encore utilisée peut être virtuellement allouée à plusieurs processus (noyau Linux).

Les programmes dans l'espace utilisateur disposent de pouvoirs restreints sur la mémoire : ils doivent demander au noyau de la mémoire. Le noyau fait appel à son gestionnaire de mémoire pour allouer (ou non) la mémoire au processus qui la demande. Si un processus tente d'utiliser des zones de mémoire ne lui appartenant pas, il est évincé automatiquement. Le mécanisme d'éviction repose sur un mécanisme du processeur, nommément une **unité de gestion de la mémoire (MMU : memory management unit)**, qui signale au noyau l'existence d'un accès fautif. C'est le noyau lui-même qui prend la décision de suspendre ou détruire immédiatement le processus fautif.

Nous avons donc vu que le système d'exploitation **dirige l'utilisation de la mémoire**. Il retient la **liste des emplacements de mémoire utilisés**, et par **qui**, ainsi que la liste des **emplacements libres**. Le système d'exploitation **réserve un emplacement de mémoire** lorsqu'un processus le demande, et le **libère** lorsqu'il n'est plus utilisé, par exemple lorsque le processus s'est arrêté.

Si un processus modifie, accidentellement ou intentionnellement un emplacement de mémoire utilisée par un autre processus, il met celui-ci en danger. S'il modifie un emplacement utilisé par le système d'exploitation il met en danger l'ensemble du système informatique.

Pour éviter un tel incident, le système d'exploitation réserve à chaque programme un e**space d'adressage (emplacement en mémoire)** que seul le programme en question peut manipuler.

La **mémoire virtuelle** permet d'exécuter simultanément plus de programmes que ce que la mémoire centrale peut contenir. Chaque programme n'ayant pas besoin que la totalité des informations qu'il manipule soit présente dans la mémoire centrale, une partie des informations est stockée dans la mémoire de masse (fichier ou partition de disque dur) habituellement plus importante mais plus lente et sont transférées en mémoire centrale lorsque le programme en a besoin.

Les programmes disposent d'un ou plusieurs espaces virtuels de mémoire continus pour travailler. Les adresses des données sont dites virtuelles dans la mesure où l'**information adressée ne se trouve pas forcément ni en mémoire centrale, ni à l'adresse indiquée**. Lorsque le programme essaie de lire ou écrire une donnée dans sa mémoire virtuelle, l'unité de gestion de mémoire cherche l'adresse physique correspondant à l'adresse virtuelle sollicitée grâce à une **table de correspondance**. Si l'emplacement n'est pas présent en mémoire centrale (on appelle cela une **faute de page**), il n'y aura évidemment aucune adresse physique correspondante. Le système d'exploitation devra alors chercher à libérer un espace en mémoire centrale en **échangeant (swap)** le contenu d'un emplacement donné de mémoire centrale avec le contenu sollicité, qui se trouve en mémoire de masse. Cette opération s'effectue **automatiquement**, à l'insu des programmes.

Des **mémoires associatives**, incorporées dans l'unité de gestion de mémoire, accélèrent le calcul des adresses. Les systèmes d'exploitation utilisent généralement deux mémoires associatives : une pour le **mode noyau** et une pour le **mode utilisateur.**

La **mémoire du mode noyau** est arrangée de manière à permettre au processeur d'utiliser la totalité de la mémoire centrale disponible lors de l'exécution des programmes du noyau du système d'exploitation.

Tandis que celle du **mode utilisateur** est arrangée de manière à **protéger le noyau** (qui est ainsi invisible pour le programme en question) lors de l'exécution des programmes hors du noyau. C'est ce que l'on nomme la **protection**, et ces mécanismes constituent les **principales caractéristiques du mode protégé.**

Chaque **programme** dispose de sa **propre table de correspondance**, ce qui permet de les isoler les uns des autres. Lors d'une commutation de contexte, le système d'exploitation placera la table du programme courant dans la mémoire associative. Le système d'exploitation crée également de nouvelles tables pour les programmes qui démarrent et décide quels emplacements de mémoire virtuelle seront ou ne seront pas présents en mémoire centrale.

3 Périphériques

Les **périphériques** sont tous les **dispositifs informatiques** qui permettent au **processeur** de **communiquer avec l'extérieur** : clavier, imprimante, carte réseau, mémoire, disque dur. Ils permettent en particulier de recevoir des informations, d'en envoyer, ainsi que de stocker des informations, les collecter dans le but de les renvoyer plus tard.

Une des responsabilités du système d'exploitation est de suivre l'état d'utilisation, libre ou réservé de tout le matériel du système informatique. Lorsqu'un matériel libre est demandé par un processus, il est alors réservé à ce processus. Pour utiliser un périphérique, le système d'exploitation se sert d'un **contrôleur** et d'un **pilote** de périphérique.

Un **contrôleur** est un **composant électronique**, qui comporte une **mémoire tampon**, et manipule un certain type de périphérique (disque dur, imprimante, mémoire, lecteur de bande magnétique...). Le contrôleur est souvent intégré au périphérique. Les différents contrôleurs disponibles sur le marché ne s'utilisent pas tous de la même manière.

Les instructions de manipulation d'une gamme de contrôleurs donnée sont incluses dans un **pilote (driver)** informatique : un l**ogiciel qui exploite les possibilités offertes par les contrôleurs**. Les pilotes informatiques font **partie du système d'exploitation**, et **offrent des services uniformes** utilisés par les autres programmes du système d'exploitation.

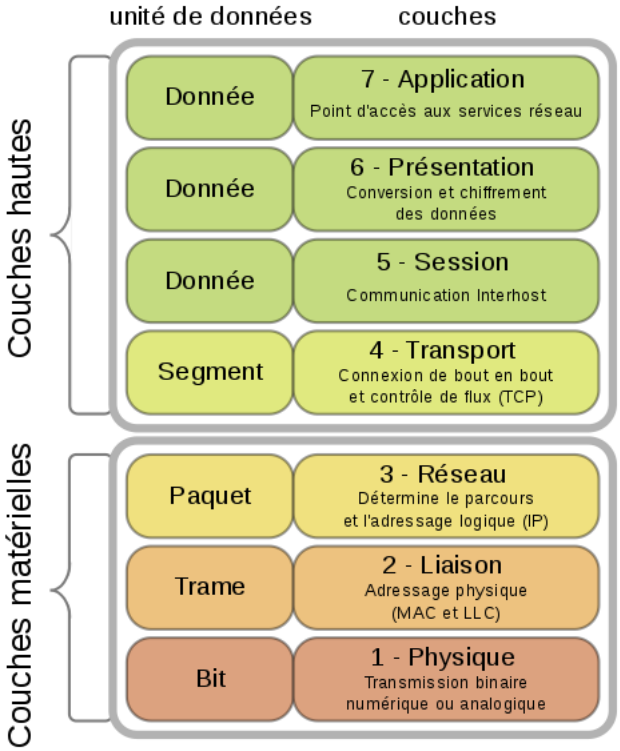
Il existe deux stratégies de manipulation des contrôleurs :

- Les contrôleurs rapides sont manipulés en **programmed I/O**. dans cette stratégie le processeur envoie des demandes d'opérations au contrôleur puis vérifie de manière intermittente l'état du contrôleur pour vérifier si l'opération demandée est terminée.
- Les contrôleurs moins rapides sont manipulés en **interrupt driven I/O**. Dans cette stratégie le processeur effectue une demande au contrôleur, puis continue d'exécuter des logiciels applicatifs. Le contrôleur envoie un signal électrique lorsque l'opération est terminée. Lors de la venue de ce signal, le processeur interrompt l'exécution des logiciels applicatifs et exécute un programme particulier **interrupt service routine** qui vérifie le nouvel état du contrôleur.

Certains périphériques ne peuvent pas être partagés, et leur utilisation est alors dédiée à un seul programme à la fois. Certains périphériques peuvent être virtuels, ou leur utilisation peut être indirecte. Par exemple l'utilisation d'une imprimante n’entraîne pas une impression immédiate parce que les informations sont tout d'abord mises en attente. Cette technique du **spool** permet l'utilisation partagée d'un périphérique qui sans ça ne pourrait pas être partagé.

4 Réseau

Dans un **réseau informatique**, **deux ordinateurs** reliés **communiquent** dès lors que les **communications** se font de part et d'autre selon les **mêmes protocoles réseau**. Selon le **modèle OSI**, les différents protocoles existants sont répartis sur sept niveaux, où un protocole d'un niveau donné peut être combiné avec n'importe quel protocole des niveaux situés en dessus et en dessous (voir encapsulation).



modèle OSI (*Open Systems Interconnection*)

Norme de communication, en réseau, de tous les systèmes informatiques.

Un **système d'exploitation** contient typiquement plusieurs programmes nécessaires pour des échanges d'informations dans différents **protocoles des niveaux 1 à 4**. Tandis que les **niveaux 5 à 7** sont pris en charge par les **logiciels applicatifs** et les **middleware**.

Pour les échanges d'informations selon les protocoles de **niveau 1 et 2**, le système d'exploitation demande l'**opération au matériel de l'ordinateur par l'intermédiaire d'un pilote informatique**, pilote qui peut faire partie intégrante du système d'exploitation ou être fourni par le constructeur du matériel.

Lors de l'envoi d'informations sur le réseau, un logiciel applicatif crée une information, la met en forme conformément aux protocoles des niveaux 7 à 5, puis la transmet au système d'exploitation. Divers programmes du système d'exploitation vont découper cette information en trames, puis vont mettre en forme les trames et les envoyer conformément aux protocoles des niveaux 4 à 1.

Lors de la réception de trames depuis le réseau, divers programmes du système d'exploitation vont tenter de les décoder conformément à différents protocoles des niveaux 1 à 4, puis transformer la

suite de trames en un flux continu, qui sera envoyé au logiciel applicatif destinataire. Le logiciel va alors décoder le flux conformément aux protocoles de niveaux 5 à 7. Le logiciel applicatif effectue préalablement une connexion, c'est-à-dire une liaison logique par laquelle il va s'associer avec un flux particulier.

Le choix exact des protocoles utilisés dépend de l'ordinateur concerné et des liaisons réseau qui vont être utilisées. Divers paramètres de configuration permettent d'influencer le choix des protocoles. Ils permettent par exemple d'empêcher l'utilisation de protocoles interdits sur le réseau concerné.

5 Variables d’environnement

Une **variable d'environnement** est une **valeur dynamique**, **chargée** en **mémoire**, pouvant être **utilisée par plusieurs processus fonctionnant simultanément**. Sur la plupart des systèmes d'exploitation, les emplacement de certaines librairies, voire des principaux exécutables du système peuvent avoir un emplacement différent selon l'installation.

Ainsi, grâce aux variables d'environnement, il est possible, à partir d'un programme, de faire référence à un emplacement en s'appuyant sur les variables d'environnement définissant ces données.

Sous Windows, les variables d'environnement sont entourées du caractère « % ». Ainsi, pour afficher la valeur d'une variable d'environnement, il suffit de taper une commande du type :

echo %NOM_DE_LA_VARIABLE%

Testez les commandes suivantes :

%USERNAME% / %USERPROFILE% / %WINDIR% / %TIME% / %OS%, etc.

Pour aller plus loin : Créez un script .bat qui va réaliser ces différentes commandes et écrire les résultats dans un fichier texte.

VIII. Annexes

Exemples de systèmes d'exploitations :

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 ; Ubuntu, Fedora, Red Hat, Debian, Archlinux, Android) et Unix (pour les serveurs ; OS X, Solaris, All Linux).

Pour les téléphones, on trouve Android, iOS (Apple), Symbian et Windows Phone.

Liste des systèmes d'exploitation : https://fr.wikipedia.org/wiki/Liste_des_syst%C3%A8mes_d%27exploitation