

Séance 1

Structure des systèmes d'exploitation



Ce(tte) œuvre est mise à disposition selon les termes de la Licence Creative Commons Attribution – Pas d'Utilisation Commerciale – Pas de Modification 4.0 International.

Objectifs

- Définitions de système informatique et **système d'exploitation**
 - Historique, architecture et exemples
 - Fonction et opérations faites par l'OS
- Décrire et comprendre les **services offerts par un OS**

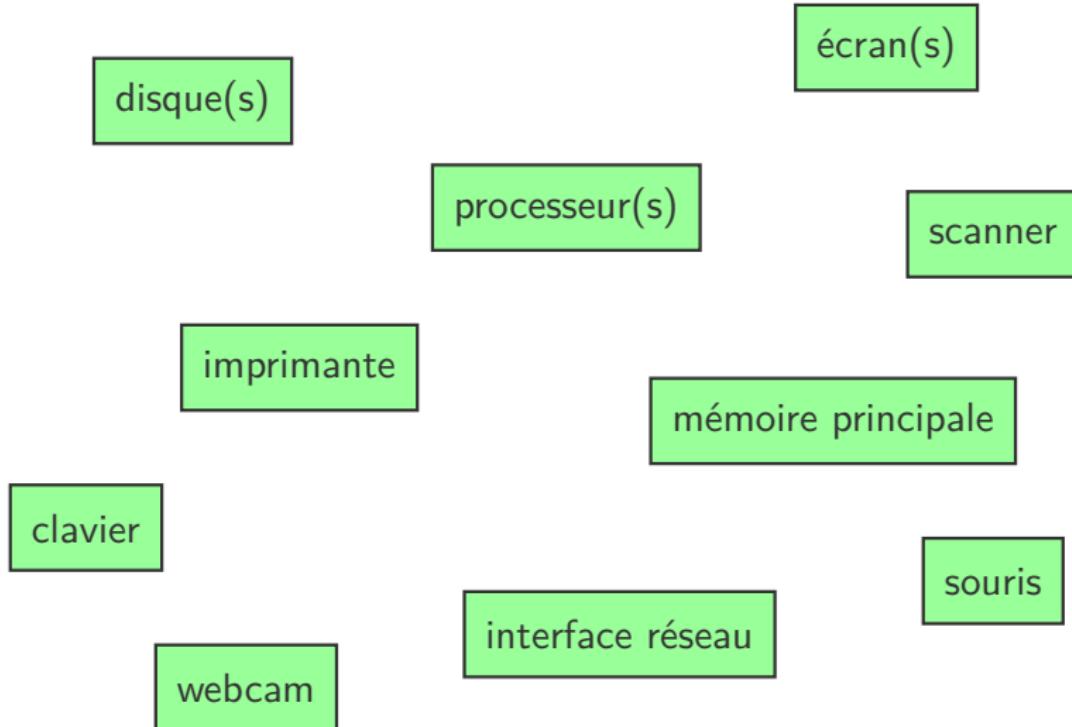
Aux utilisateurs, aux processus et aux autres systèmes
- Comprendre les **structures** possibles d'un OS

Types de structure, comparaison et cas d'OS existants

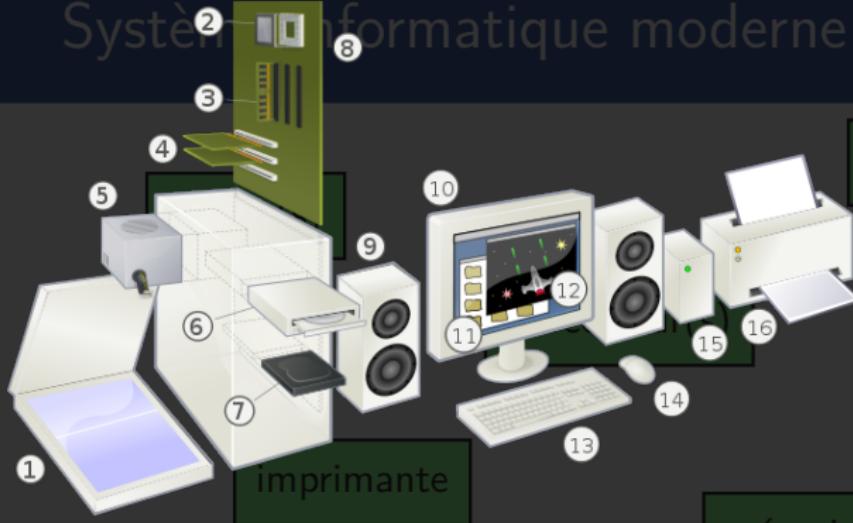
Système informatique



Système informatique moderne



Système informatique moderne



interface



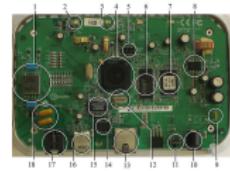
Évolution des systèmes informatiques

- Expérience d'**automatisation** (machines de calcul)
- Système **spécifique** (fixed-purpose computer)
 - Armée (casser des codes, calculer des trajectoires)
 - Gouvernement (calcul du salaire des fonctionnaires)
- Système **générique** (general purpose computer)
Naissance des systèmes d'exploitation

Type de système informatique

■ Plusieurs types de systèmes informatiques

- PC (personal computer)
- Mainframe
- Station de travail (workstation)
- Système mobile (portable, tablette, smartphone...)
- Système embarqué



Environnement informatique (1)

- Les OS sont utilisés différemment selon l'**environnement**

Aspects spécifiques aux différents types d'environnement

- Informatique **traditionnelle**

PCs connectés au réseau, serveurs (services de fichier/impression)

- Informatique **mobile** portable et léger

Smartphone, tablettes

- Systèmes **distribués**

Systèmes physiquement séparés mis en réseau {L, W, M, P}AN

Environnement informatique (2)

- Informatique **client-server**

Serveurs répondent aux requêtes de clients

- Informatique **peer-to-peer** (P2P)

Tous les nœuds du système distribué sont des pairs

- **Virtualisation**

Permet à un OS de s'exécuter sur un autre OS

- **Autres** types d'environnement

Cloud computing, informatique embarquée temps-réel...

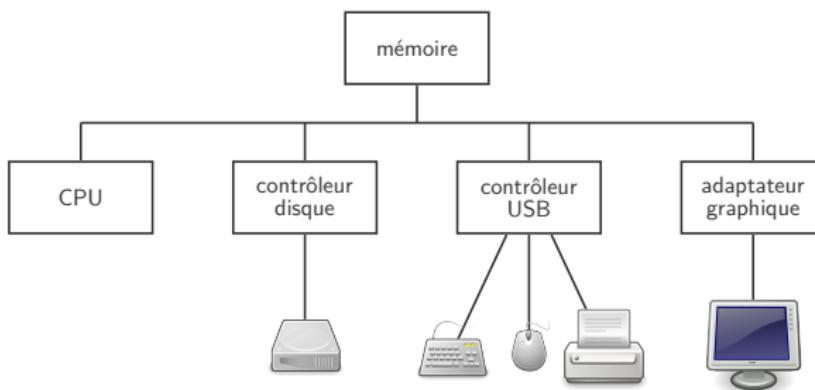
Vue générale d'un système informatique

- Un ou plusieurs CPU et des **contrôleurs de périphérique**

Connectés à une mémoire partagée par un bus

- Compétition CPU/contrôleurs pour des **cycles mémoire**

Un contrôleur mémoire pour synchroniser les accès



Architecture des systèmes informatiques

- Système avec **un seul processeur** générique principal
 - Exécution des processus systèmes et utilisateurs
 - Aide de processeurs spécifiques (disque, graphique, E/S...)
- Système **multi-processeurs** (parallèle ou multi-cœurs)
 - Plusieurs processeurs en communication proche
 - Partage du bus et parfois horloge, mémoire et périphériques
- Système **en clusters**
 - Plusieurs systèmes individuels (nœuds) liés entre eux
 - Plusieurs processeurs faiblement couplés

Système informatique vs d'exploitation

- Système informatique composé de trois éléments
 - Matériel (hardware)
 - Logiciel (software)
 - Données
- L'OS fournit un environnement pour utiliser ces ressources

Pas de fonction utile en soit, utilisé par les autres programmes
- OS modernes dirigés par les interruptions

Code OS exécuté suite à des évènements signalés par interruption

Amorçage de la machine

- Nécessite un **programme initial** pour amorcer la machine
Bootstrap, stocké en ROM ou EEPROM (Bios, Firmware)
- **Initialisation** de plusieurs aspects du système
Registres CPU, contrôleur périphérique, contenu de la mémoire...
- Localise puis charge en mémoire le **noyau de l'OS** (kernel)
Boot sector

Processus système

- Services additionnels hors noyau, **processus systèmes** (démon)
« init » est le premier processus système sous Unix
- Exécutés durant **toute la durée** d'exécution du kernel
Le démon « init » en lance d'autres sous Unix
- **Démarrage complet** du système une fois les démons lancés
Le système attend l'occurrence d'évènements

Gestionnaire d'évènements

- Occurrence d'un évènement signalée par une **interruption**
 - Hardware par l'envoi d'un signal vers le CPU
 - Software par l'exécution d'un appel système
- Transfert du contrôle vers un **gestionnaire d'interruption**

Le CPU arrête son exécution courante et sauve son état
- Gestionnaire générique suivi d'un **gestionnaire spécifique**

Vecteur d'interruption pour stocker les adresses des gestionnaires

Système d'exploitation

FLATRON E240

Missing operating system.

Exemples de systèmes d'exploitation



android



ubuntu



macOS



VxWorks

- Système d'exploitation temps réel (RTOS)
 - Système embarqué performances temps réel et déterministe
 - Certification de sûreté et de sécurité
- Lancé en 1987 et dernière release en mars 2014
 - Intel (x86, Quark SoC, x86-64), MIPS, PowerPC, SH-4, ARM
 - VxWorks 5 premier RTOS avec support réseau dans les '90
 - VxWorks 7 sorti en 2016 avec support pour l'IoT



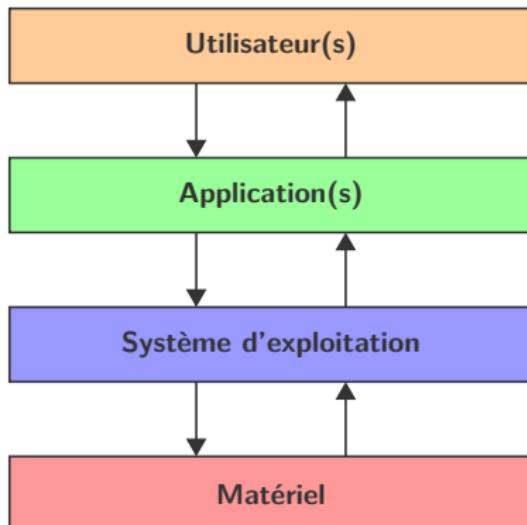
Système d'exploitation

- Couche logicielle intermédiaire **entre l'utilisateur et le hardware**
Fournit aux programmes utilisateurs une interface avec le matériel
- **Trois objectifs** principaux
 - **Pratique** : utilisation aisée par les utilisateurs (PC, mobile)
 - **Efficace** : optimiser l'utilisation du hardware (mainframes)
 - **Évolution** : ajout de nouvelles fonctions systèmes
- Système logiciel très large et **très complexe**
Créé pièce par pièce avec une délimitation claire

Structure d'un système informatique (1)

- Système informatique peut être vu comme quatre couches

Utilisation de services inférieurs pour en offrir des supérieurs



Structure d'un système informatique (2)

- Le **matériel** fournit les ressources de calcul

Processeur (CPU), mémoire, périphériques d'entrée-sortie (E/S)

- Le **système d'exploitation** coordonne l'utilisation du matériel

Coordination entre les applications et utilisateurs

- Les **applications** résolvent les problèmes des utilisateurs

Traitement de texte, compilateur, jeux vidéos, navigateurs web...

- Les **utilisateurs** utilisent le système informatique

Personnes, machines, autres ordinateurs

Abstraction proposée par l'OS

- **Environnement** permettant aux applications de travailler

En fournissant un accès à l'hardware aux applications

- Plusieurs **abstractions** sont proposées par l'OS

Manipulation transparente du hardware

Matériel	Abstraction
CPU et mémoire	Processus
Stockage sur disque	Fichier
Interface réseau	Socket, RPC, RMI
Affichage	Librairies de dessin, fenêtres

Point de vue utilisateur de l'OS

- La **vue utilisateur** dépend de l'interface utilisée

Classiquement assis derrière un PC avec écran, clavier et souris

- Plusieurs **types d'environnement** possibles
 - **Desktop** : pratique et facile à utiliser
 - **Mainframe** : maximiser utilisation et partage des ressources
 - **Station de travail** : compromis usability/utilisation ressources
 - **Système portable** : ressources, utilisation, batterie
 - **Système embarqué** : pas de vue utilisateur

Point de vue système de l'OS

- L'OS est le programme le plus intimement **lié au hardware**

Une partie de son travail sert à gérer le système

- Deux principales activités en lien avec **le système**

- **Allocation des ressources**

*Partage efficace et équitable, gestion des conflits du temps CPU
espace mémoire, espace disque, périphériques E/S...*

- **Programme de contrôle**

Protège des erreurs et des mauvaises utilisations de l'ordinateur

Définition d'un système d'exploitation

- Il n'y a **pas de définition** universellement adoptée

Grande diversité : toaster, voiture, navette spatiale, maison, machine de jeu, système de contrôle industriel...

- « *Tout ce que contient la boîte lorsque vous achetez un OS* »

Procès Microsoft en 1998 avec Internet Explorer

- « *Seul programme qui tourne tout le temps sur l'ordinateur* »

Kernel, programmes systèmes et programmes d'applications

Le procès Microsoft (1)

- Département de la Justice et 19 états américains **vs** Microsoft

Procès antitrust sans jury (juge seul) ouvert le 19 octobre 1998

- **Concurrence logicielle** de type paradigmatique selon Microsoft

- Développement d'un logiciel et mise sur le marché
- Entreprise imitée par un ensemble de concurrents
- Le meilleur produit emporte la part du lion

- Violente bataille concurrentielle puis **monopole de fait**

Ébranlé que par l'apparition d'un nouveau paradigme

Le procès Microsoft (2)

- Microsoft gagne plusieurs **batailles paradigmatiques**
 - 1981 : lancement de MS-DOS, adopté par IBM
 - 1985 : lancement de Windows
 - Windows 95 pour répondre au Macintosh d'Apple
 - Windows 98 et Internet Explorer
- Apparition de **Netscape Navigator** en 1994, puis Sun Java
 - Compatibles avec n'importe quel système d'exploitation*
- **Monopole** et blocage d'innovations (5 novembre 1999)
 - Marché des OS pour ordinateurs avec un processeur Intel*



Bref historique

Processing en série

- **Interaction directe** avec le hardware (fin '40–mi-'50)

Pas de présence de système d'exploitation

- **Commandes physiques** pour utiliser le système

Lumière, switches, système d'entrée et imprimante

- Deux grands **problèmes d'utilisation** de ces systèmes

- **Ordonnancement** : réserver le temps machine avec une sheet
 - **Temps de setup** : plusieurs jeux de cartes pour exécuter job

Ordinateur central

- Les **mainframes** ont des grosses capacités d'E/S

Gestion optimale de plusieurs jobs à la fois

- Trois **types de service**

- **Batch** : statistiques de ventes dans un magasin
- **Transactionnel** : réservation de billets d'avion
- **Temps partagé** : plusieurs utilisateurs

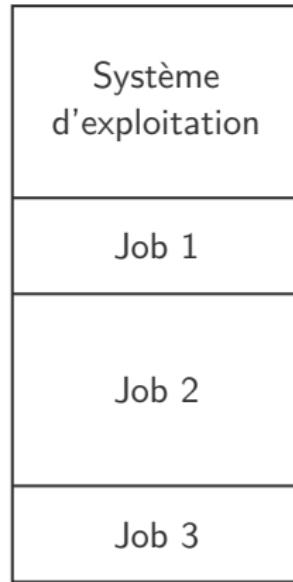
- Premier OS rudimentaire appelé *monitor*

- Un opérateur batche les jobs ensemble
- Séquençage des jobs : monitor → job → monitor

Multiprogrammation (1)



Système simple par lot



Traitement par lot multiprogrammé

Multiprogrammation (2)

- Plusieurs éléments à gérer dûs à la **multiprogrammation**
 - Attribuer les ressources
 - Gérer la mémoire allouée à plusieurs tâches
 - Planifier l'utilisation du CPU, donc les tâches à exécuter
- Le CPU est **multiplexé** entre plusieurs tâches

La tâche doit être en mémoire (swap avec le disque)
- **Partage** des ressources et du temps de calcul
 - Plusieurs batch de jobs avec multiprogrammation
 - Plusieurs utilisateurs avec time-sharing

Ordinateur personnel

- Système informatique dédié à **un seul utilisateur**

Avec périphériques d'E/S (clavier, souris, écran, imprimante...)

- Offrir une **interface conviviale**

Commode et réactive

- **Exemples** d'OS

Linux, macOS, UNIX, Windows...

Système parallèle

- Plusieurs CPU sont en communication proche

Partagent le bus, parfois horloge, mémoire et périphériques E/S

- Trois avantages principaux

- Augmentation du débit ($< N$ avec N processeurs)
- Économie d'échelle par partage de ressources
- Augmentation de la fiabilité (des processeurs peuvent cracher)

- Dégradation douce du système

Service fourni \propto Hardware survivant

Multiprocessing

- Multi-processeur **symétrique** (SMP)
 - Tous les processeurs sont égaux
 - Plusieurs processus exécutés en même temps
 - Supporté par la plupart des OS modernes
- Multi-processeur **asymétrique** (AMP)
 - Une tâche spécifique par processeur
 - Un processeur maître contrôle le système
- Processeur **multicœur**

Profite de la communication plus rapide intra-puce

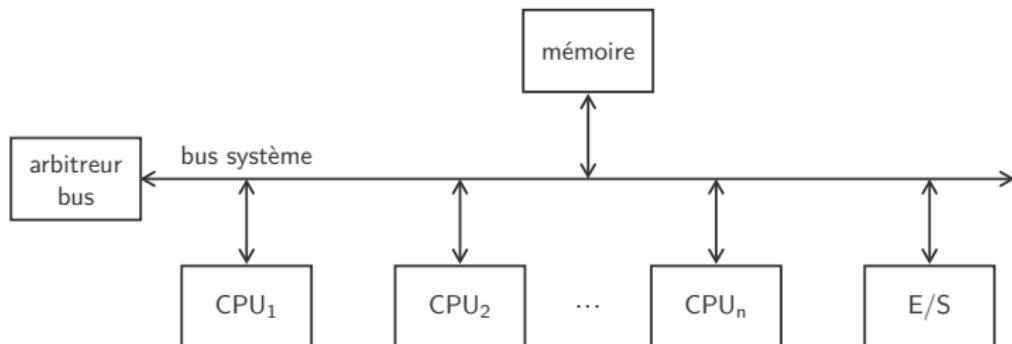
Multiprocessing symétrique

- Chaque CPU dispose de **sa propre cache**

Permet d'avoir du stockage local pour l'efficacité

- Les CPU partagent la **mémoire principale**

Permet de partager des informations et communiquer entre CPU



Système distribué

- Rassemble **plusieurs processeurs** physiques
- Contrairement aux multiprocesseurs, ici **faiblement couplé**
Via le réseau local (LAN) ou un lien plus rapide (InfiniBand)
- Principaux **avantages**
 - Partage de ressources
 - Calcul de haute performance : partage de charge
 - Fiabilité
- Nécessite une **infrastructure réseau**

Système temps-réel

- Typiquement utilisé dans des **applications de contrôle**
- Contrainte forte en **hard real-time**
 - Le temps de réaction est fixé
 - La mémoire secondaire est limitée ou absente
- Contrainte souple en **soft real-time**
 - Un temps maximal de réaction est fixé
 - Applications multimédias, réalité virtuelle...

Système portable

- Smartphones, tablettes...
- **Contraintes** et problèmes
 - Mémoire limitée
 - Processeurs lents
 - Petits écrans d'affichage
 - Consommation

Tolérance aux pannes

- Un OS doit avoir une bonne **tolérance aux pannes**

Continuer normalement après des pannes hardware/software

- Augmentation de la **fiabilité** d'un système

- Ajout d'un certain degré de redondance
- Augmentation des couts financiers ou de performance

- Trois **mesures de base** de la tolérance aux pannes

Fiabilité, temps moyen avant panne et disponibilité

Mesure de la tolérance aux pannes

- Désir d'un **fonctionnement correct** d'un système

Exécution de programme, protection des données vs modifications

- **Fiabilité** $R(t)$

Probabilité de fonctionner au temps t s'il fonctionnait en $t = 0$

- Mesures de deux **temps moyens** par rapport aux pannes

- **Avant panne** : $MTTF = \int_0^{\infty} R(t)$

- **Avant réparation** : $MTTR$

Disponibilité

- Un système peut être **disponible ou non**

Temps indisponible est downtime, et disponible est uptime

- Disponibilité mesure fraction de temps où **système répond**

$$A = \frac{MTTF}{MTTF + MTTR}$$

Classe	Disponibilité	Downtime annuel
Continu	1.0	0
Tolérance aux pannes	0.99999	5 minutes
Résistant aux pannes	0.9999	53 minutes
Haute disponibilité	0.999	8.3 heures
Disponibilité normale	0.99–0.995	44–87 heures

Services offerts par l'OS

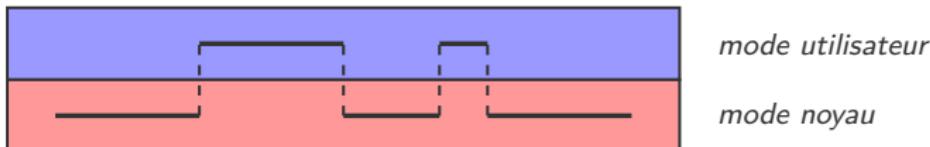
S E R V I C E

Fonctions de l'OS

- L'OS agit comme un fournisseur de **services**
Système de fichiers, librairies standards, système de fenêtres
- **Coordinateur** sur trois aspects
 - **Protection** : les jobs ne doivent pas interférer entre eux
 - **Communication** : les jobs doivent pouvoir interagir entre eux
 - **Gestion des ressources** : facilite le partage de ressource

Opérations faites par l'OS

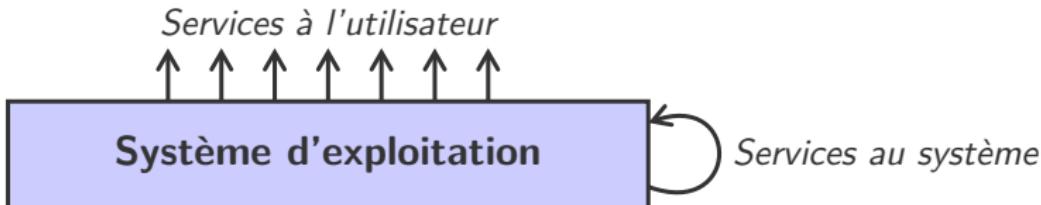
- Tant qu'il n'y a rien à faire, l'**OS ne fait rien** de particulier
Processus à exécuter, E/S à traiter, répondre aux utilisateurs
- Alternance entre deux **modes d'exécution**
 - Distinguer l'exécution du code de l'OS et de l'utilisateur
 - Support hardware pour différencier les modes
- Instructions désignées comme **privilégiées**
Ne peuvent être exécutées qu'en mode noyau



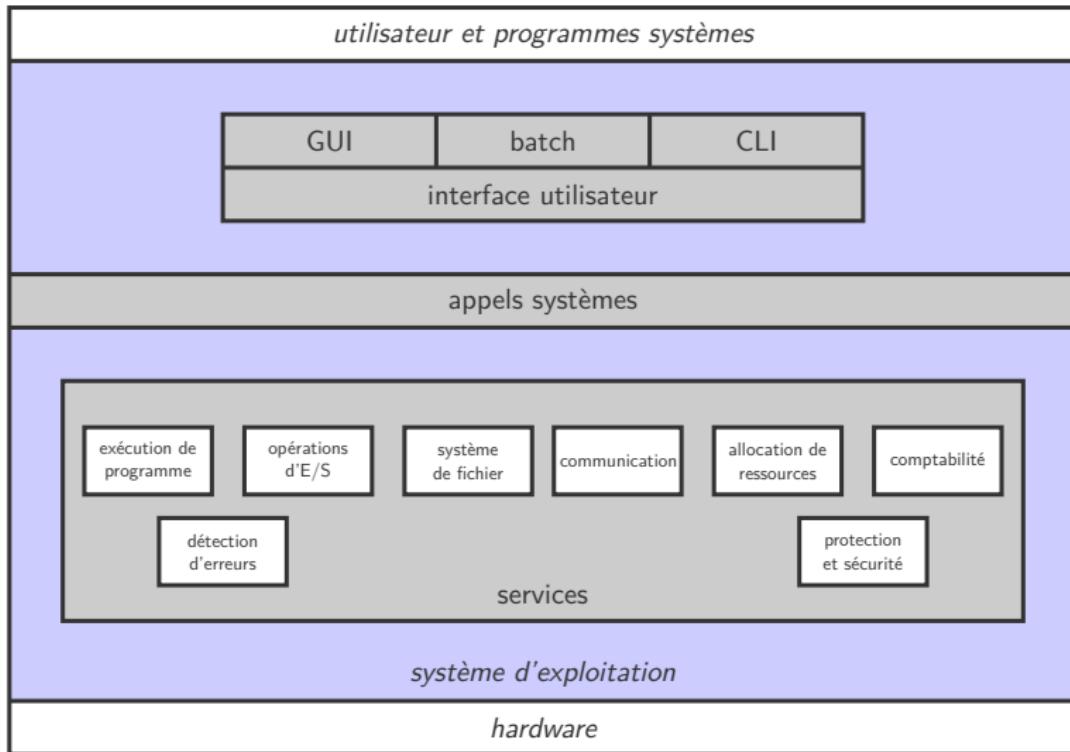
Services offerts par l'OS (1)

- L'OS propose deux types de **services**
 - À l'utilisateur
 - Au système
- Les services offerts **varient** selon le système d'exploitation

Des classes de service communes sont identifiables



Services offerts par l'OS (2)



Service à l'utilisateur (1)

- Interface utilisateur (UI) peut prendre plusieurs formes
Ligne de commande (CLI), batch, interface graphique (GUI)
- Chargement et exécution d'un programme
Chargement en mémoire, terminaison normale ou en erreur
- Opérations d'entrées/sorties (E/S)
Depuis/vers un fichier, un périphérique...
- Manipulation du système de fichiers
Écrire/lire, propriétés, permissions, propriétaire...

Service à l'utilisateur (2)

- **Communication** entre processus

Sur la même machine ou à travers le réseau

- Mémoire partagée
- Passage de messages

- **Détection d'erreurs** et (tentative de) correction

Par exemple, plus de papier dans l'imprimante

- Dans CPU, mémoire hardware, périphériques E/S, disque, réseau, programme utilisateur...
- Action à prendre par l'OS pour chaque type d'erreur

Service au système

- Allocation de ressources aux utilisateurs et jobs

Cycles CPU, mémoire principale, espace de stockage

- Code spécifique : cycle CPU, mémoire principale...
- Code générique : E/S, périphérique de stockage USB...

- Comptabilisation pour statistiques ou facturation

Ressources comme temps CPU par exemple

- Protection et sécurité

- Accès contrôlé aux ressources du système
- Authentification des utilisateurs

Interface en ligne de commande (CLI)

- Permet d'entrer directement des **commandes**

À faire exécuter par le système d'exploitation

- Plusieurs **implémentations** possibles

- Directement inclus dans le noyau (MS-DOS)
- Comme un programme système (Windows, shells Unix)

- **Fonctionnement** similaire des interpréteurs

- Attend une commande de l'utilisateur
- Deux manières d'exécuter la commande

Implémentées par l'interpréteur ou comme programme système

Shell

- Interpréteur de commandes parfois appelés **shell**

Notamment sur les OS avec plusieurs interpréteurs

- Permet d'accéder à des **services** offerts par l'OS

- Commandes exécutées par le shell (un seul exécutable)
 - Un programme pour chaque commande (plusieurs exécutables)

- Fonctionne dans une **boucle REPL** (read-eval-print loop)

Attente de commande, évaluation puis affichage du résultat

Interface graphique (GUI)

- Interface de bureau conviviale
 - Clavier, écran et souris
 - Fenêtres et menus, icônes
- Première GUI sur le **Xerox Alto Computer** (1973)
- La plupart des systèmes incluent **CLI et GUI**
 - Windows : GUI avec l'interpréteur MS-DOS
 - macOS : interface « Aqua » avec un kernel Unix
 - Solaris, Linux : CLI avec GUI optionnelle (Gnome, KDE...)

Appel système

- Utilisation des services de l'OS via des **appels systèmes**
 - Interface vers les services mis à disposition par l'OS
 - Généralement accessible comme des routines C/C++
- Tâches de **très bas niveau** à écrire en langage d'assemblage

Par exemple pour faire un accès direct au hardware

- **Application Programming Interface (API)**
 - Encapsule séquence d'appels systèmes pour éviter appel direct
 - Par exemple : Win32, POSIX, Java...
 - Accès à une API à partir d'une librairie (par exemple libc)

Exemple : Copie d'un fichier (1)

- 1 Récupérer le nom du fichier source
 - Afficher une phrase sur la sortie standard
 - Lire l'entrée standard
- 2 Récupérer le nom du fichier destination
 - Afficher une phrase sur la sortie standard
 - Lire l'entrée standard
- 3 Ouvrir le fichier source, créer et ouvrir le fichier destination
- 4 Tant que la lecture n'échoue pas
 - Lire un caractère depuis la source
 - Écrire le caractère dans la destination
- 5 Fermer les fichiers source et destination

Exécution d'un appel système (1)

- Passage du mode utilisateur au **mode noyau**

L'appel système est exécuté dans un mode privilégié de l'OS

- Gestion via une **interface des appels systèmes**

- Fais le relais en interceptant les appels systèmes dans l'API
- Utilise une table associative numérotant les appels systèmes

- Le **programme utilisateur** ignore les détails de l'appel système

- Il doit respecter l'API et passer les bons paramètres
- Il doit comprendre la valeur de retour

L'appel système read

```
#include <unistd.h>
ssize_t read(int fd, void *buf, size_t count);
```

■ Paramètres

- int fd : descripteur de fichier
- void *buf : buffer où les données lues seront placées
- size_t count : le nombre maximal d'octets à lire

■ Valeur de retour

- le nombre d'octets lus en cas de succès
- 0 si la fin du fichier est atteinte
- -1 en cas d'erreur

Aide des appels systèmes (1)

READ(2) BSD System Calls Manual READ(2)

NAME
pread, read, ready -- read input

LIBRARY
Standard C Library (`libc, -lc`)

SYNOPSIS

```
#include <sys/types.h>
#include <sys/uio.h>
#include <unistd.h>

ssize_t
pread(int d, void *buf, size_t nbytes, off_t offset);

ssize_t
read(int filedes, void *buf, size_t nbytes);

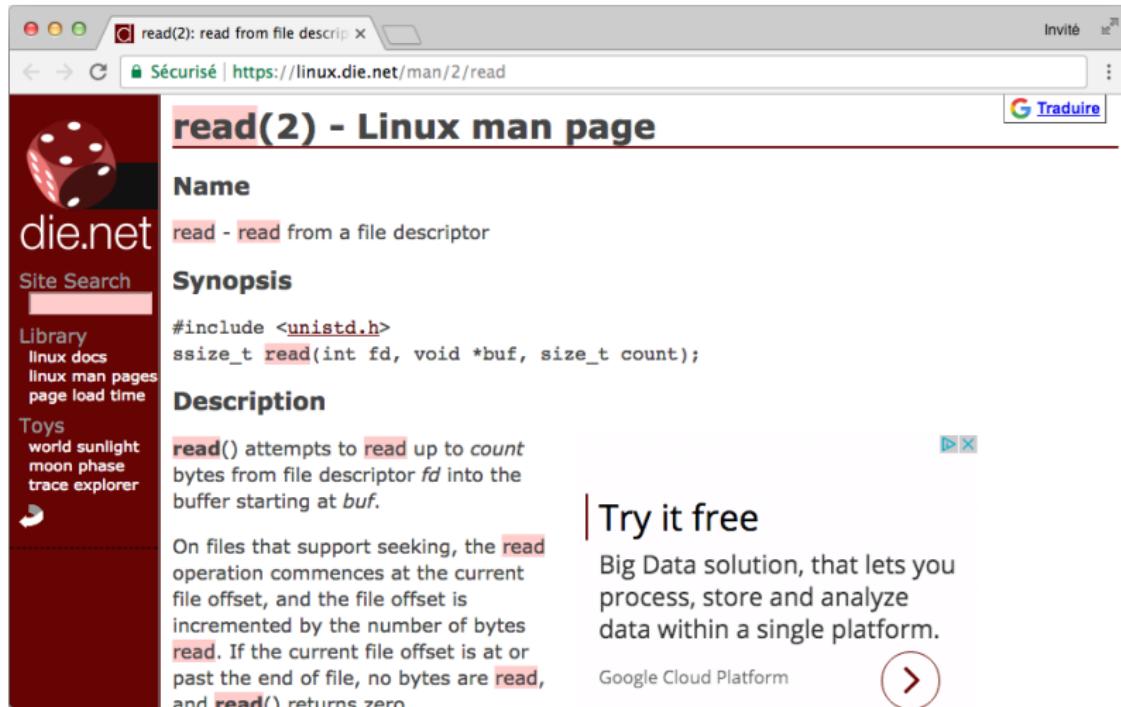
ssize_t
readv(int d, const struct iovec *iov, int iovcnt);
```

DESCRIPTION
`Read()` attempts to read `nbytes` bytes of data from the object referenced by the descriptor `filedes` into the buffer pointed to by `buf`. `Ready()` performs the same action, but scatters the input

:

Aide obtenue avec la commande « `man 2 read` » (appels systèmes en section 2)

Aide des appels systèmes (2)



The screenshot shows a web browser window displaying the Linux man page for the `read` system call. The URL is <https://linux.die.net/man/2/read>. The page has a header "read(2) - Linux man page". It includes sections for "Name", "Synopsis", and "Description". The "Synopsis" section contains the following C code:

```
#include <unistd.h>
ssize_t read(int fd, void *buf, size_t count);
```

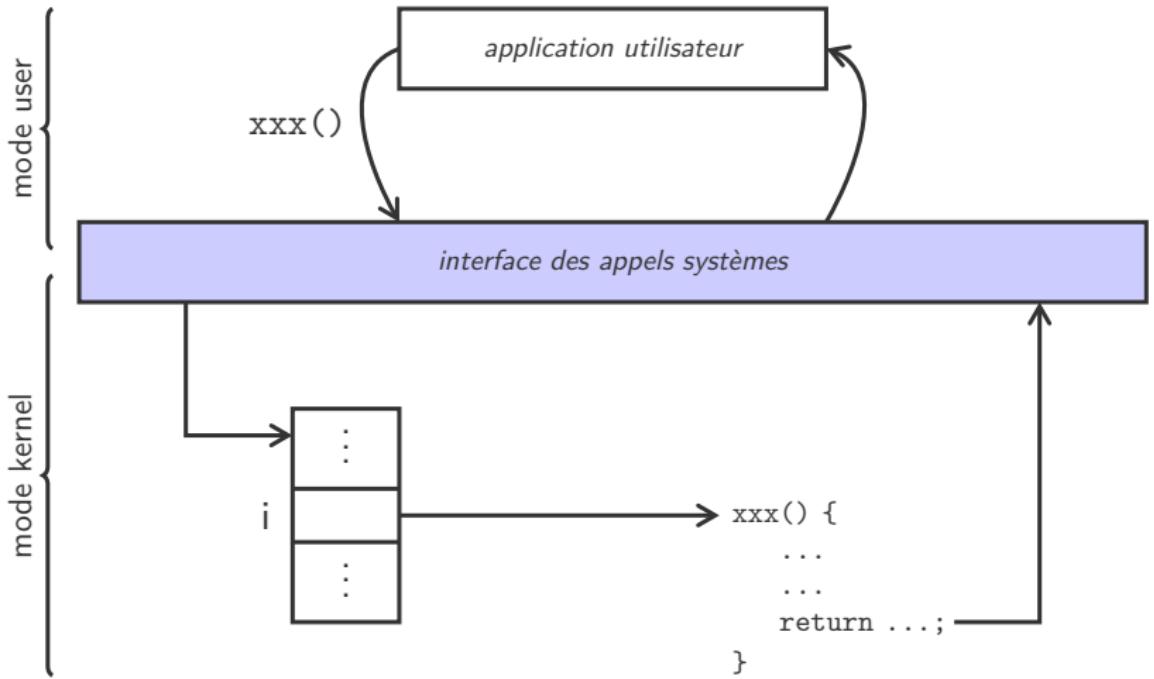
The "Description" section states:

`read()` attempts to `read` up to `count` bytes from file descriptor `fd` into the buffer starting at `buf`.

On files that support seeking, the `read` operation commences at the current file offset, and the file offset is incremented by the number of bytes `read`. If the current file offset is at or past the end of file, no bytes are `read`, and `read()` returns zero.

On the right side of the page, there is a "Try it free" section for Google Cloud Platform, featuring a "Get Started" button.

Exécution d'un appel système (2)



Exemple : Copie d'un fichier (2)

- Version simplifiée de la **copie d'un fichier**

Aucune gestion des erreurs

```
1 void copy (char *src, char*dst)
2 {
3     char buffer[BUFFSIZE];
4     int srcFile = open (src, O_RDONLY);
5     int dstFile = open (dst, O_WRONLY | O_CREAT, 0666);
6
7     if (srcFile != -1 && dstFile != -1)
8     {
9         int r;
10        while ((r = read (srcFile, buffer, BUFFSIZE)))
11        {
12            write (dstFile, buffer, r);
13        }
14
15        close (srcFile);
16        close (dstFile);
17    }
18 }
```

Types d'appels systèmes

- **Appels systèmes** disponibles dépend du système d'exploitation
Plusieurs appels sont communs à la plupart des OS
- **Six catégories** principales d'appels systèmes
 - Contrôle des processus
 - Gestion des fichiers
 - Gestion des périphériques
 - Maintenance de l'informatique
 - Communication
 - Protection

Programme système

- Interfe avec l'OS à l'aide de **programmes systèmes**

Utilitaire système aide développement/exécution de programme

- Plusieurs **catégories** de programme système

- Gestion de fichiers : ls, mv, mkdir, rm...
- Information d'état : whoami, time, ps, top...
- Modification de fichiers texte : vi, nano...
- Support de langages de programmation : cc, java...
- Chargement et exécution de programme : gdb, bash...
- Communication : ftp, mail, ssh...
- Services d'arrière plan : bg, screen, tmux...

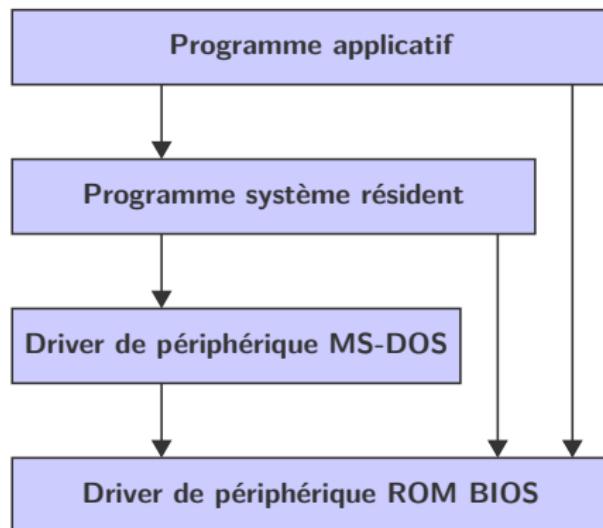
Structure des OS



Structure simple

- MS-DOS écrit en assembleur 8088
 - Logiciel monolithique, pas divisé en modules
 - Maximum de fonctionnalités avec le moins d'espace
 - Pas de séparation entre les différentes couches de fonctionnalité
- Pas de protection hardware ni de mode d'exécution
 - MS-DOS initialement développé pour Intel 8088
 - Tous les programmes ont accès à l'ensemble des ressources

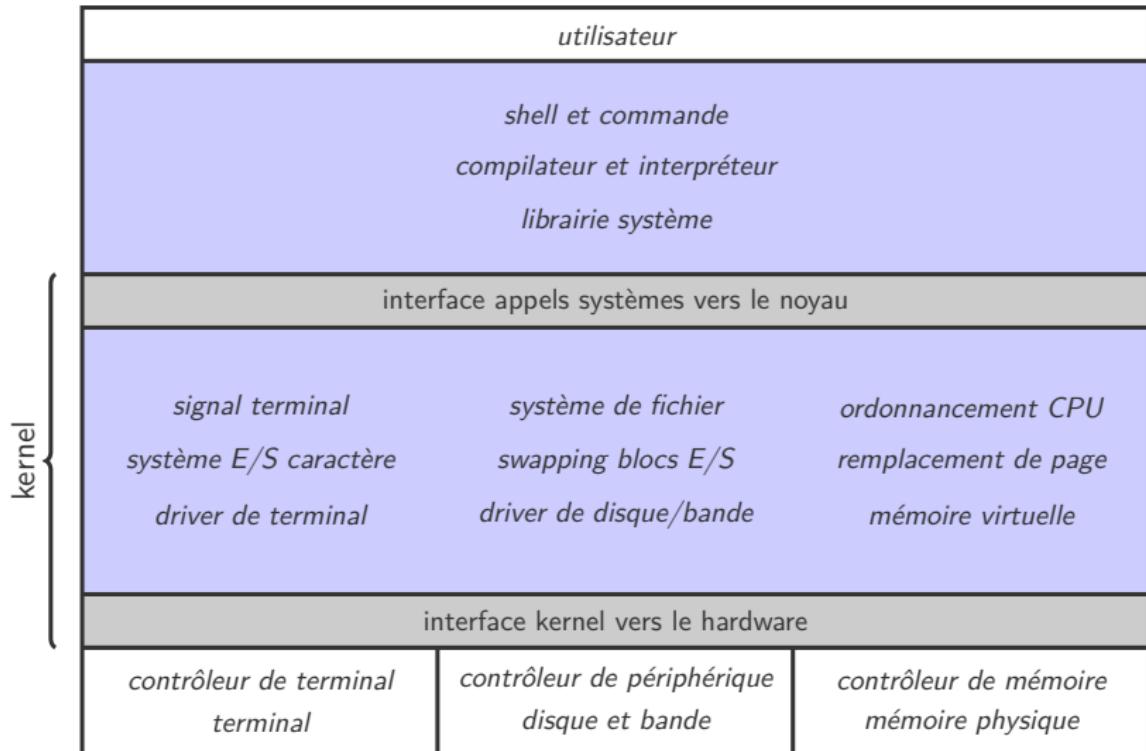
Structure en couches de MS-DOS



Structure limitée

- **UNIX original** en deux parties séparables
 - Noyau avec interfaces et drivers de périphérique
 - Programme système (shell, compilateur, commande...)
- **Noyau** réside entre interface appels systèmes et hardware
 - Noyau monolithique avec beaucoup de fonctionnalités
 - Plutôt difficile à implémenter et à maintenir
 - Bonne performance avec peu d'overhead de communication

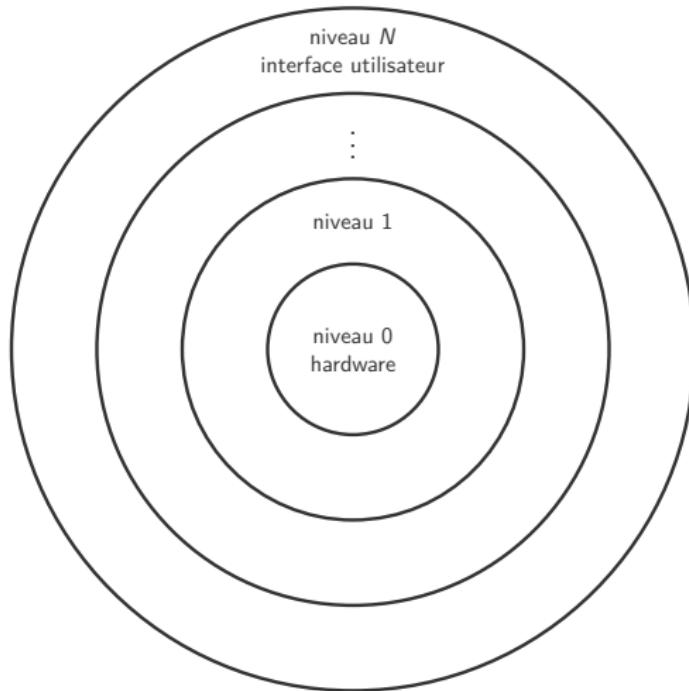
Structure en couches de UNIX



Approche en niveaux (1)

- **Support hardware** pour cloisonner des parties du système
L'OS peut mieux contrôler le système et les applications l'utilisant
- L'OS est découpé en niveaux
 - Niveau 0 : hardware
 - ...
 - Niveau N : interface utilisateur
- Chaque niveau est **construit sur le niveau inférieur**
Il ne doit pas savoir comment les autres sont implémentés

Approche en niveaux (2)



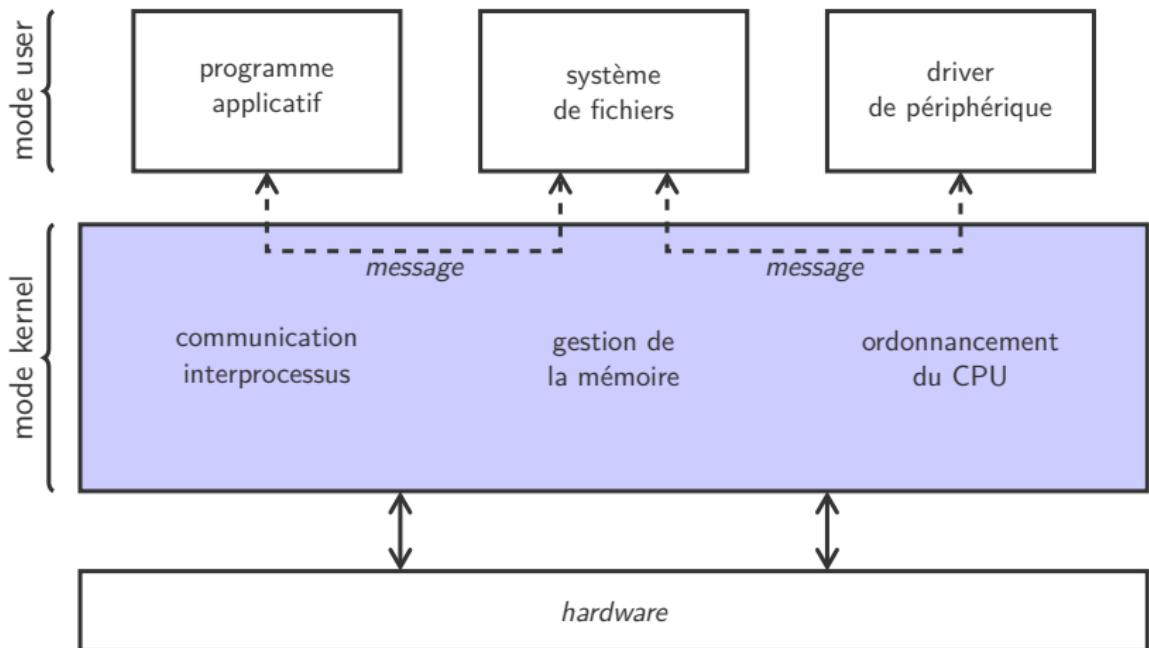
Approche en niveaux (3)

- Plusieurs **avantages** de la découpe en niveaux
 - Liberté d'implémentation des niveaux pour le développeur
 - Simplicité de construction et debugging, modularité
 - Possibilité de cacher de l'information avec interface
- Quelques **inconvénients** majeurs
 - Définition des niveaux pas unique et évidente
 - Moins efficace par lourdeur d'appels entre niveaux

Micro-noyau (1)

- Taille du noyau **UNIX** augmente avec fonctionnalités
 - Le noyau devient gros et difficile à maintenir*
- Ne garder que **le minimum** dans le noyau
 - Le reste est déplacé en programmes système ou utilisateur*
- Quelques services souvent gérés par le **micro-noyau**
 - Gestion minimale de processus et de mémoire
 - Outil de communication (passage de messages)
 - Gestion des interruptions (hardware)

Micro-noyau (2)



Micro-noyau (3)

- Série d'**avantages** liés à la petite taille
 - Extension et portage sur une autre plateforme facile
 - Plus grande sécurité et fiabilité
 - Meilleure robustesse lors d'un crash d'un service user
- Quelques **inconvénients** majeurs
 - Surcharge et explosion du nombre de fonctions systèmes
 - Lourdeur due aux communications

Mach

- Système d'exploitation **Mach** développé en mi-'80 à CMU
 - Modularisation du noyau avec l'approche micro-noyau*
- **Utilisé dans** plusieurs systèmes d'exploitation
 - Tru64 UNIX fournit interface UNIX à l'utilisateur
 - Mapping des appels systèmes UNIX en messages vers user-service*
 - Noyau Darwin de macOS

Module (1)

- Développement de **modules noyaux** chargeables

Composants cœurs et liens vers services additionnels par modules

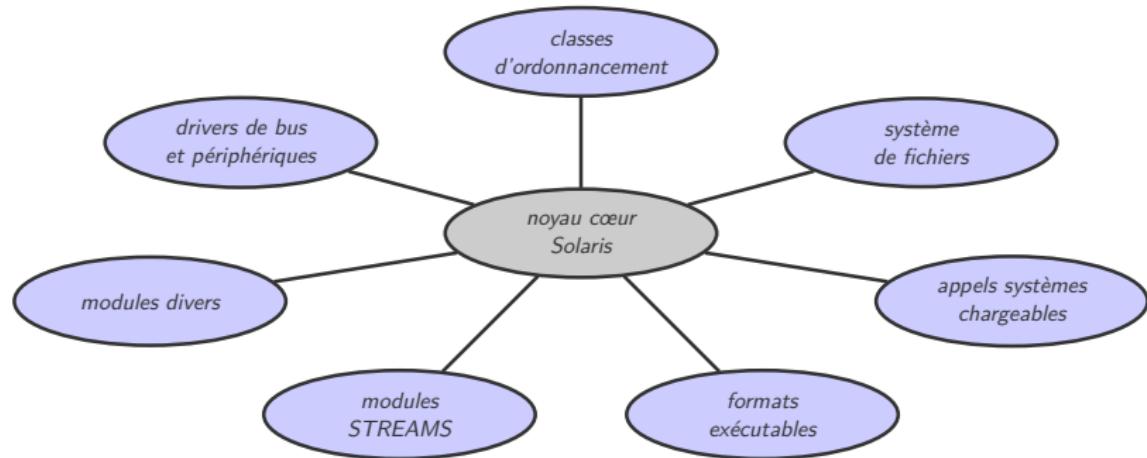
- Simplification de l'**extensibilité** des fonctionnalités

- Chargement de modules au démarrage ou à l'exécution
- Évite de devoir recompiler le kernel pour chaque ajout

- Points communs avec niveaux et micro-noyau et plus **flexible**

- Modules sont séparés avec interfaces connues
- Chargés dans le noyau lorsque nécessaire, minimum par défaut
- Communication directe sans passage de messages

Module (2)

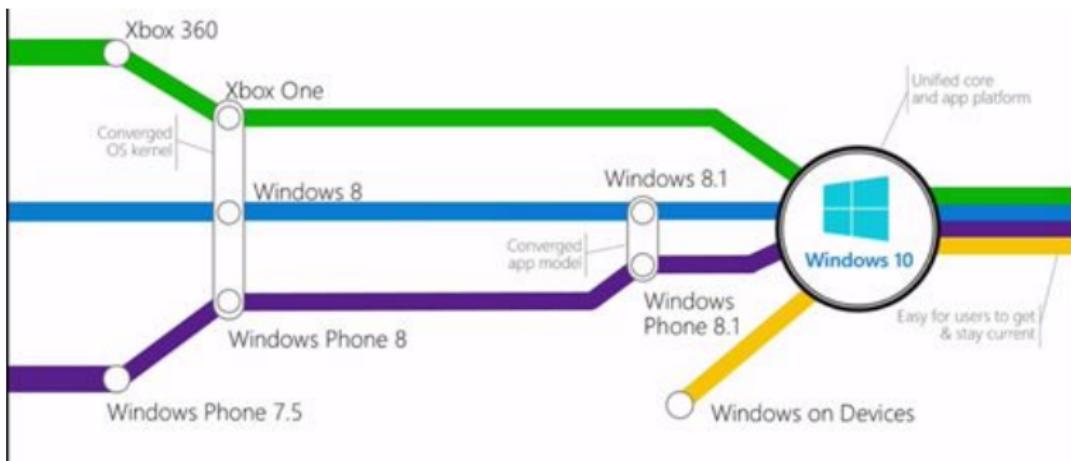


Système hybride

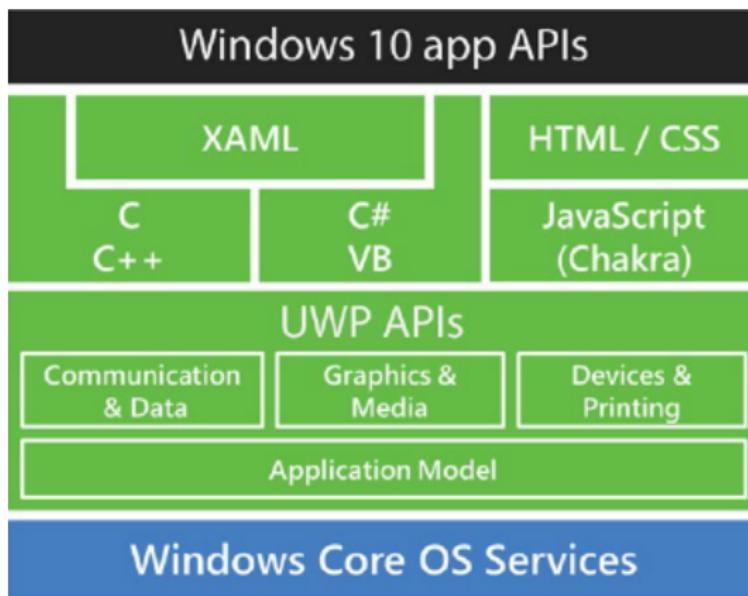
- Combinaison de **différentes structures** dans un même système

Pour penser aux performances, à la sécurité et l'utilisabilité
- Exemples de **systèmes communs**
 - **Linux et Solaris** : monolithique dans l'espace noyau, modulaire pour le chargement dynamique de modules
 - **Windows** : monolithique, orientation micro-noyau pour différents sous-systèmes
 - **macOS et iOS** : en niveaux, GUI Aqua, env. de prog. Cocoa, micro-noyau Mach, kernel UNIX BSD en-dessous
 - **Android** : en niveaux, avec un kernel Linux modifié en bas

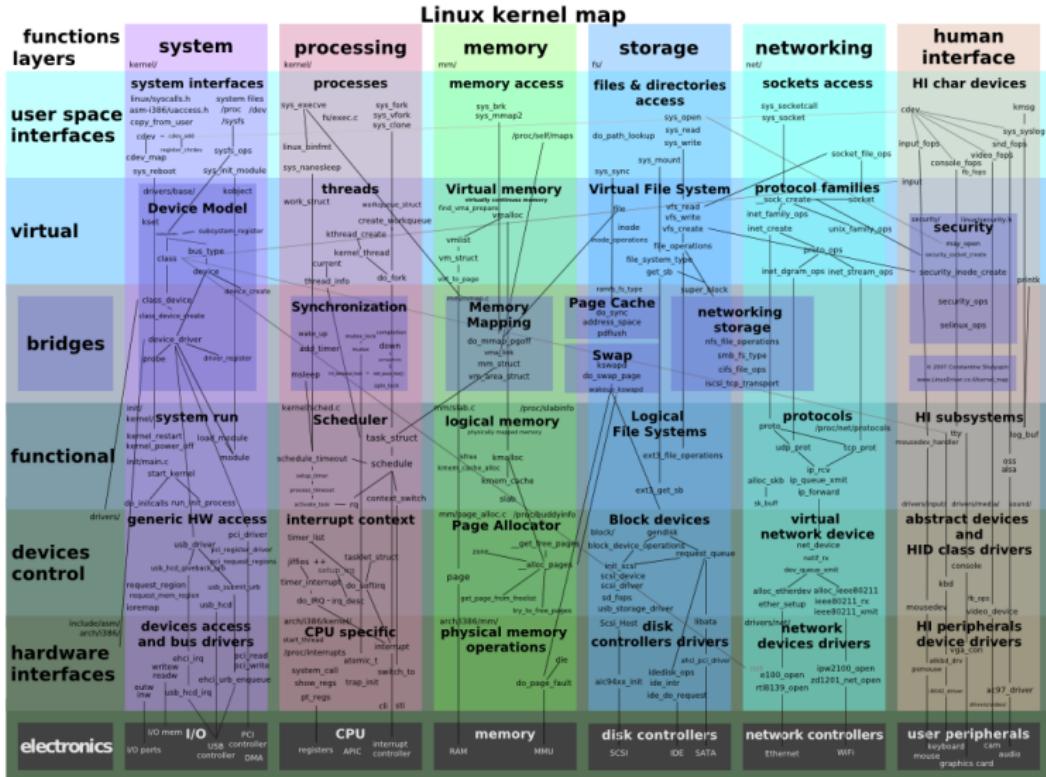
Architecture de Windows 10 (1)



Architecture de Windows 10 (2)



Architecture du kernel Linux



Architecture de macOS

macOS Architecture

Cocoa Application – Application User Interface responds to user events , manages app behavior.

App Kit	Notification Center	Siri	Sharing	Full Screen Mode	Cocoa Autolayout	Popovers	Software Configuration	Accessibility	Apple Script	Spotlight
---------	---------------------	------	---------	------------------	------------------	----------	------------------------	---------------	--------------	-----------

Media – Plays , records , editing audio-visual media , rendering 2D and 3D graphics.

AV Foundation Audio playback, editing, analysis & recording.	Core Animation 2D rendering & animation. 3D Transformations.	Core Audio Audio services for recording, playback and synchronization.	Core Image Fast image processing. Uses GPU based acceleration.	Core Text Handles Unicode fonts & texts.	Open AL Delivers 3D audio. High performance parallel playbacks in games.	Metal Portable 3D graphics apps using imaging functions & effects.	Quartz macOS graphics, rendering support for 2D content. Event routing & cursor management.
---	--	---	---	---	---	---	--

Core Services – Fundamental Services for low level network communication , Automatic Reference Counting , Data Formatting , String Manipulation.

Address Book Centralized database for contacts & groups.	Core Foundation Declares C based programmatic interfaces. Data Types & Data Management.	Quick Look Enables Spotlight & Finder to display thumbnail images.	Security User authentication, certificates & keys, authorization, keychain services etc.	Core Data Data model management & storage, undo/redo, validation of property values.	Foundation Swift framework for object behavior, internationalization, data types & data mgmt.	Social Supports integration with social networking services.	Webkit Display HTML content in apps. Contains WebCore and JavaScript Core.
---	--	---	---	---	--	---	---

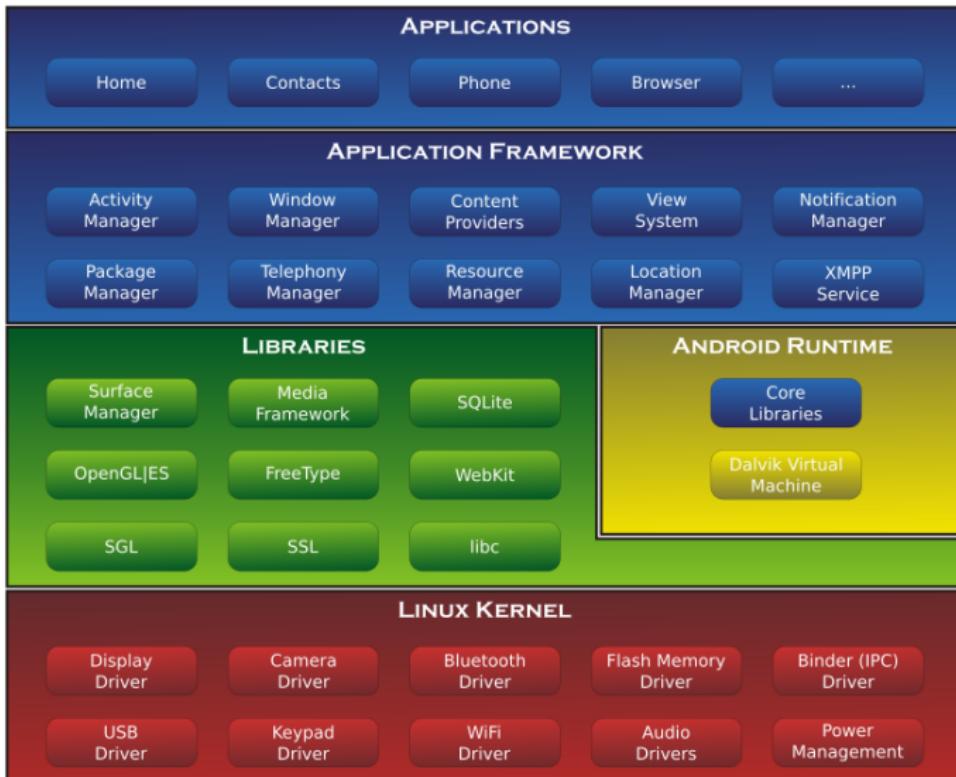
Core OS – Related to hardware and networking. Interfaces for running high-performance computation tasks on CPU or GPU.

Accelerate Accelerate complex operations, improve performance using vector unit, supports data parallelism, 3D graphic imaging, image processing.	Directory Services Provides access to collected information about users, groups, computers, printers in a networked environment.	Disk Arbitration Notifies when local or remote volumes are mounted and unmounted.	Metal Makes the high-performance parallel processing power of GPUs available to general purpose computing.	System Configuration Provides access to current network configuration information. Determines reachability of remote hosts. Notifies about changes in network.
--	---	--	---	---

Kernel & Device Drivers – Device drivers & BSD Libraries , low level components. Support for file system security , interprocess communications.

BSD Provides basis for file systems and networking facilities, POSIX thread support, BSD sockets,	File System Supports multiple volume formats (NTFS, ExFAT, FAT, HFS+, APFS etc.) & file protocols (AFP, NFS etc.)	Mach Protected Memory, Preemptive Multitasking, Advanced Virtual Memory, Real Time Support.	Networking Supports network kernel extensions (NKEs). Create network modules, configure protocol stacks. Monitor and modify network traffic.
--	--	--	---

Architecture d'Android



Crédits (1)

- <https://www.flickr.com/photos/synapticism/21760898530>
- http://commons.wikimedia.org/wiki/File:Personal_computer,_exploded_6.svg
- http://commons.wikimedia.org/wiki/File:PS4_Console-wDS4.jpg
- http://commons.wikimedia.org/wiki/File:Boeing_787-8_N787BA_cockpit.jpg
- <http://commons.wikimedia.org/wiki/File:Snackomatic.jpg>
- https://en.wikipedia.org/wiki/File:Desktop_personal_computer.jpg
- https://en.wikipedia.org/wiki/File:MSI_Laptop_computer.jpg
- https://en.wikipedia.org/wiki/File:Front_Z9_2094.jpg
- https://en.wikipedia.org/wiki/File:Blackberry_Z10.jpg
- https://en.wikipedia.org/wiki/File:ADSL_modem_router_internals_labeled.jpg
- <https://openclipart.org/detail/34897/tango-printer-by-warszawianka>
- <https://openclipart.org/detail/34567/tango-input-mouse-by-warszawianka>
- <https://openclipart.org/detail/34561/tango-input-keyboard-by-warszawianka>
- <https://openclipart.org/detail/34903/tango-video-display-by-warszawianka>
- <https://openclipart.org/detail/34537/tango-drive-hard-disk-by-warszawianka>
- <https://www.flickr.com/photos/hinkelstone/7050697671>
- [https://en.wikipedia.org/wiki/File:Mars_%27Curiosity%27_Rover,_Spacecraft_Assembly_Facility,_Pasadena,_California_\(2011\).jpg](https://en.wikipedia.org/wiki/File:Mars_%27Curiosity%27_Rover,_Spacecraft_Assembly_Facility,_Pasadena,_California_(2011).jpg)
- https://en.wikipedia.org/wiki/File:ASIMO_4.28.11.jpg
- https://en.wikipedia.org/wiki/File:AH-64_Apache_extraction_exercise.jpg
- <https://www.flickr.com/photos/icathing/335756720>

Crédits (2)

- <https://www.flickr.com/photos/dskley/14711793077>
- <https://www.flickr.com/photos/zigazou76/7670875192>
- <http://www.c-sharpcorner.com/UploadFile/1ff178/architecture-of-windows-10/Images/kal1.jpg>
- <https://i.stack.imgur.com/PnPvr.png>
- https://en.wikipedia.org/wiki/File:Linux_kernel_map.png
- https://en.wikipedia.org/wiki/File:MacOS_Architecture_v2.svg
- <https://en.wikipedia.org/wiki/File:Android-System-Architecture.svg>