

# UE : Système d'exploitation

Licence 3 SFA Informatique



# L'UE Système d'Exploitation

Le contenu :

- 21 heures de cours / TD
- 6 heures de TP

Notation :

- TP notés
- Exam

# Objectifs

Dans ce cours seront abordés :

- Le “Système d’exploitation” en général
- Plus dans les détails sur GNU/Linux
- De l’administration système Linux
- Les TD et TP seront effectués sur Linux

Limiter la théorie pour se focaliser sur la pratique

- Mais les premiers cours seront assez théoriques...

# Qui suis-je ?

Damien Grandi

*Développeur Full Stack*

[damien.grandi@gmail.com](mailto:damien.grandi@gmail.com)



# Présentation

Et vous ?

- Votre parcours
- Êtes-vous à l'aise en anglais ?
- Quels systèmes d'exploitation connaissez-vous ?
- Quels systèmes d'exploitation avez-vous déjà utilisés ?
- Sur quel système d'exploitation êtes-vous actuellement ?

- Qu'est ce qu'un Système d'Exploitation ?
- Quelques OS connus
- Histoire des OS libres
- Les rôles et les contextes d'utilisation
- Structure d'un OS
  - Structure générale
  - Types de noyau
  - Les pilotes et les modules
  - Interruptions
  - Autorisation et arbitrage
  - Appels systèmes
  - Modes d'exécution
  - Entrées / sorties
  - Ordonnanceur
- Amorçage
- Installation d'un OS

Qu'est ce qu'un  
système  
d'exploitation ?

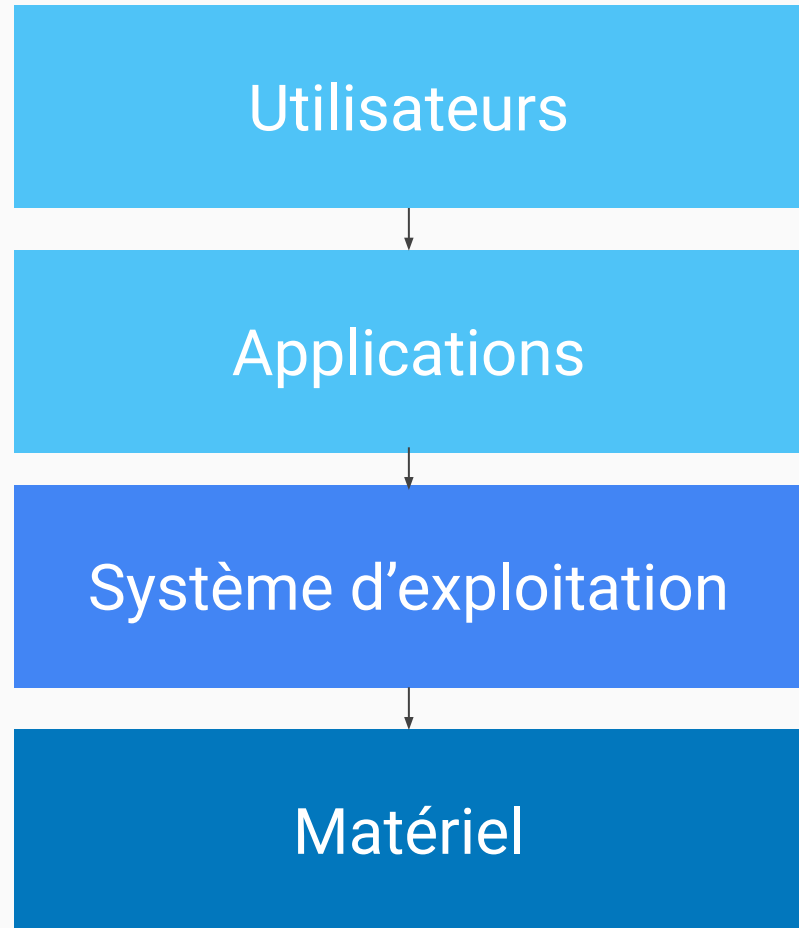
# Qu'est ce qu'un système d'exploitation ?

Un **système d'exploitation** (OS pour Operating System) est un ensemble de **programmes** qui dirige l'utilisation des ressources d'un ordinateur par des logiciels applicatifs. Il reçoit des demandes d'utilisation des ressources de l'ordinateur de la part des logiciels applicatifs.

Un **système d'exploitation** fait **l'intermédiaire** entre **l'utilisateur** et le **matériel**.



Qu'est ce qu'un  
système  
d'exploitation ?



# Matériel

Exemple de matériel  
dans le cas d'un  
ordinateur format tour



# Quelques OS connus

## Ordinateur

- Windows
- GNU/Linux
- MacOS

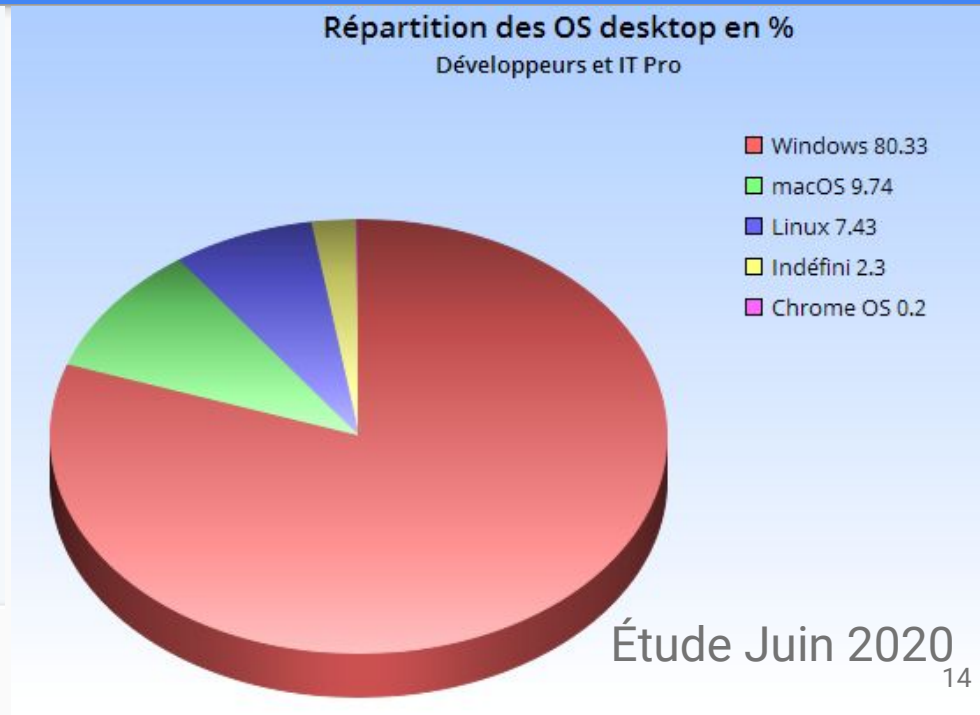
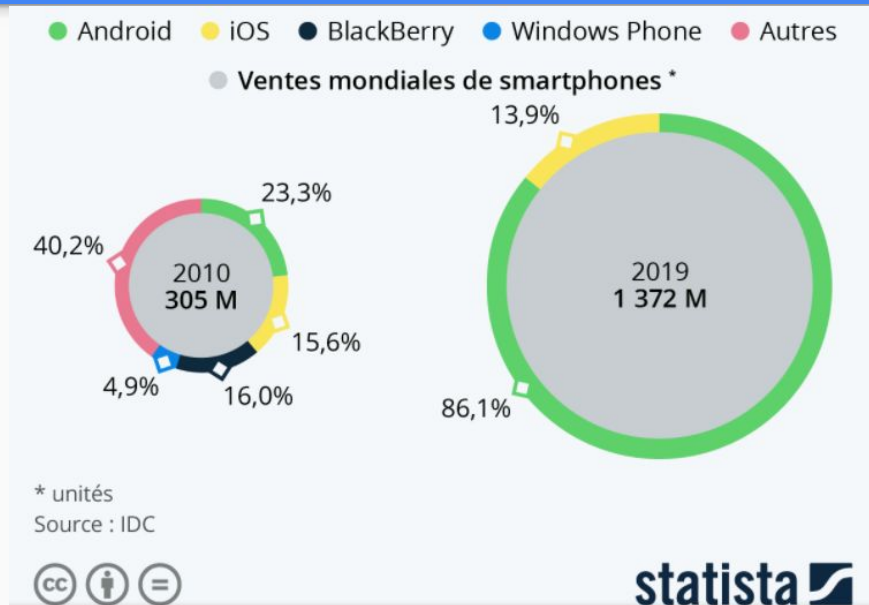
## Smartphone

- iOS
- Android

Vous en  
connaissiez  
d'autres ?

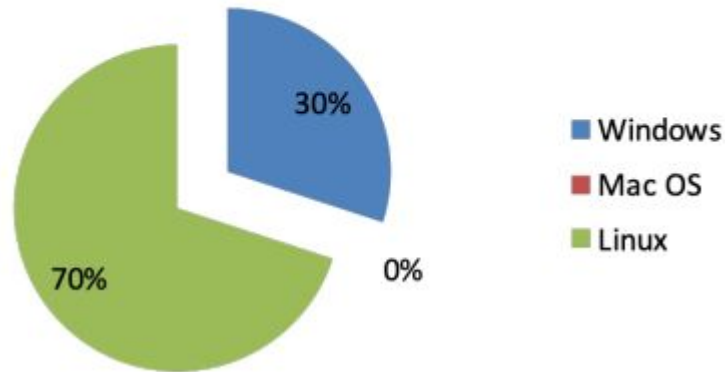
Quels sont les  
plus utilisés selon  
vous ?

# Les OS connus dans le grand public

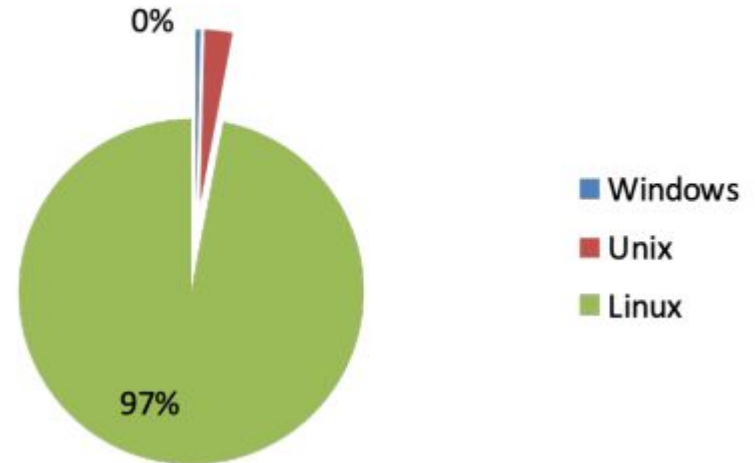


# Les OS dans le monde des serveurs

Part de marché Serveur



Part de marché Supercalculateur



# La famille Windows

- MS-DOS (Juillet 1981)
- [...]
- Windows 3.1
- Windows 95
- Windows 98
- Windows Me
- Windows XP
- Windows Vista
- Windows 7
- Windows 10



Logo de Windows de  
1985 à 1990



Logo de Windows de  
1990 à 1995



Logo de Windows de  
1995 à 2000



Logo de Windows de  
2001 à 2006. Sur  
Windows XP.



Logo de Windows de  
2006 à 2012



Logo de Windows de  
2012 à 2015.

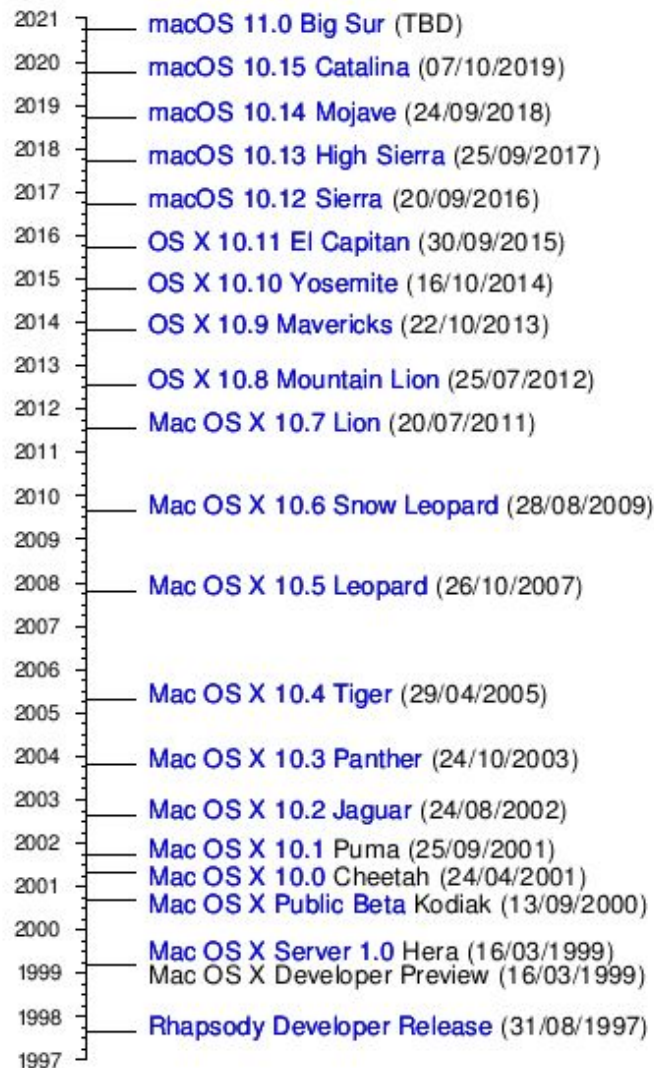


Logo de Windows depuis  
2015.



# La famille MacOS

"Système 1" en Janvier 1984

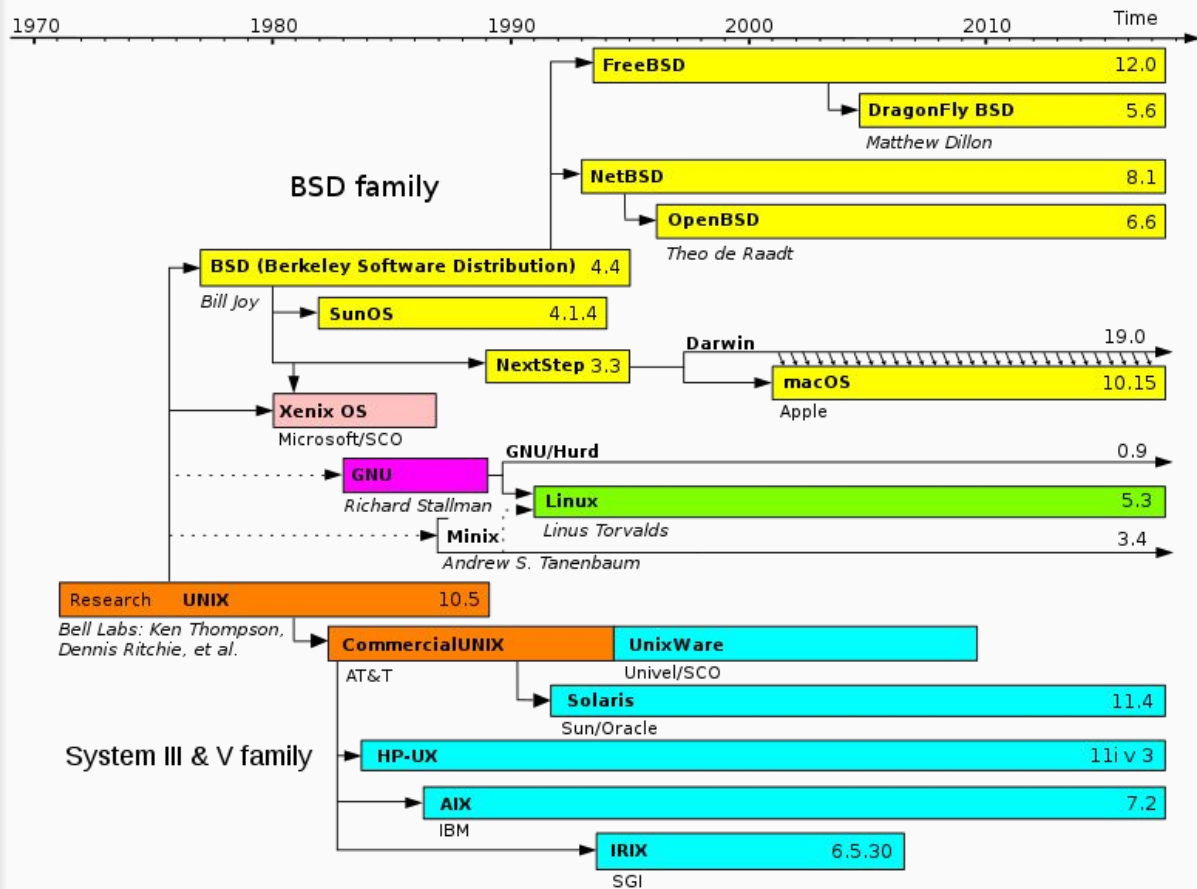


A vertical timeline on the right side of the slide, showing the progression of MacOS versions from 1997 to 2021. The years are listed on the left, and the version names and release dates are listed on the right, connected by horizontal lines. The versions are: Rhapsody Developer Release (1997), Mac OS X Server 1.0 Hera (1999), Mac OS X Developer Preview (1999), Mac OS X Public Beta Kodiak (2000), Mac OS X 10.0 Cheetah (2001), Mac OS X 10.1 Puma (2002), Mac OS X 10.2 Jaguar (2003), Mac OS X 10.3 Panther (2004), Mac OS X 10.4 Tiger (2005), Mac OS X 10.5 Leopard (2007), Mac OS X 10.6 Snow Leopard (2009), Mac OS X 10.7 Lion (2011), OS X 10.8 Mountain Lion (2012), OS X 10.9 Mavericks (2013), OS X 10.10 Yosemite (2014), OS X 10.11 El Capitan (2015), macOS 10.12 Sierra (2016), macOS 10.13 High Sierra (2017), macOS 10.14 Mojave (2018), macOS 10.15 Catalina (2019), and macOS 11.0 Big Sur (TBD) (2021).

2021	macOS 11.0 Big Sur (TBD)
2020	macOS 10.15 Catalina (07/10/2019)
2019	macOS 10.14 Mojave (24/09/2018)
2018	macOS 10.13 High Sierra (25/09/2017)
2017	macOS 10.12 Sierra (20/09/2016)
2016	OS X 10.11 El Capitan (30/09/2015)
2015	OS X 10.10 Yosemite (16/10/2014)
2014	OS X 10.9 Mavericks (22/10/2013)
2013	OS X 10.8 Mountain Lion (25/07/2012)
2012	Mac OS X 10.7 Lion (20/07/2011)
2011	
2010	Mac OS X 10.6 Snow Leopard (28/08/2009)
2009	
2008	Mac OS X 10.5 Leopard (26/10/2007)
2007	
2006	
2005	Mac OS X 10.4 Tiger (29/04/2005)
2004	Mac OS X 10.3 Panther (24/10/2003)
2003	Mac OS X 10.2 Jaguar (24/08/2002)
2002	Mac OS X 10.1 Puma (25/09/2001)
2001	Mac OS X 10.0 Cheetah (24/04/2001)
2000	Mac OS X Public Beta Kodiak (13/09/2000)
1999	Mac OS X Server 1.0 Hera (16/03/1999)
1999	Mac OS X Developer Preview (16/03/1999)
1998	
1997	Rhapsody Developer Release (31/08/1997)

# La famille MacOS

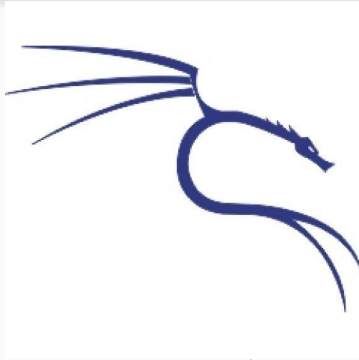
MacOS est basé sur UNIX



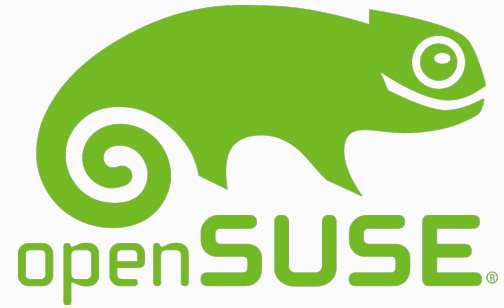
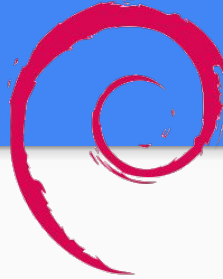
# La famille GNU/Linux



gentoo linux™



debian



Red Hat



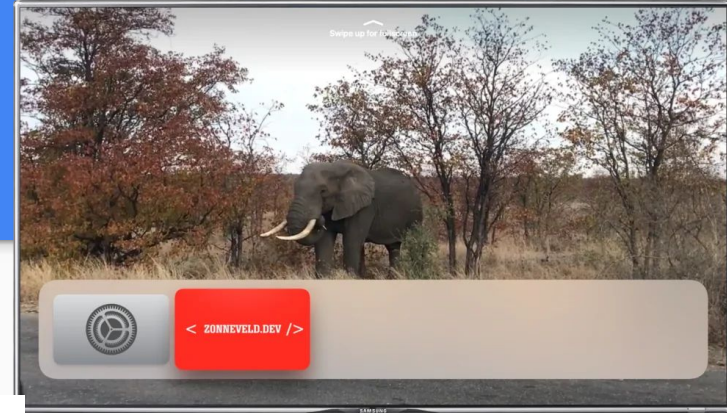
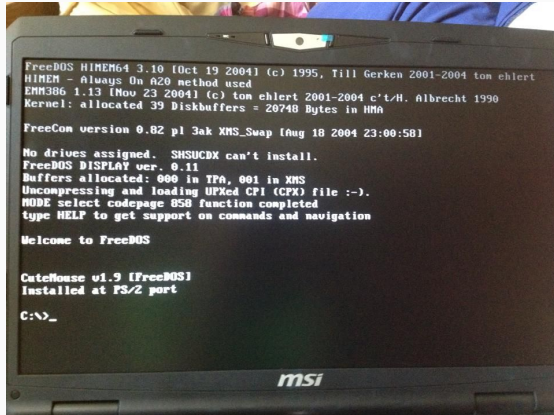
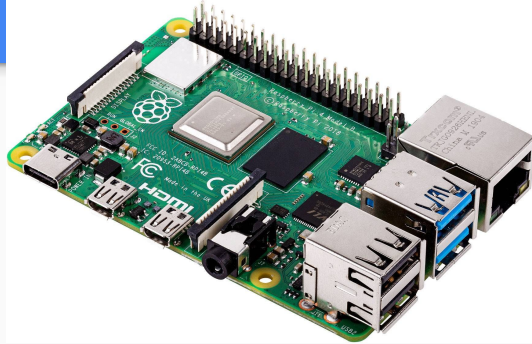
GNU en 1983  
GNU/Linux en 1991

[https://en.wikipedia.org/wiki/List\\_of\\_Linux\\_distributions](https://en.wikipedia.org/wiki/List_of_Linux_distributions)

# UNIX

[https://upload.wikimedia.org/wikipedia/commons/7/77/Unix\\_history-simple.svg](https://upload.wikimedia.org/wikipedia/commons/7/77/Unix_history-simple.svg)

# Des OS ?



Qu'est ce qui  
différencie OS de  
“pas OS” ?

# Différence OS / pas OS

OS :

- Abstraction de la complexité matérielle
- Permet l'exécution de différents programmes sans nécessiter de "re-coder" la partie qui manipule le matériel

Pas OS :

- Un programme dédié, directement exécuté, souvent amené à interagir directement avec le matériel
- Très souvent le cas sur un microcontrôleur (exception: FreeRTOS)

# Quelques différences entre OS connues des développeurs

- Caractères de saut de ligne CR LF
  - Carriage Return `\r`
  - Line Feed `\n`
- Spécificités de plateforme
  - Types de données spécifiques
    - `typedef unsigned long DWORD`
  - Différence de comportement
    - `wchar_t` (W pour Wide) taille différente



# Histoire des OS libres

# De MULTICS à UNIX : Depuis 1964

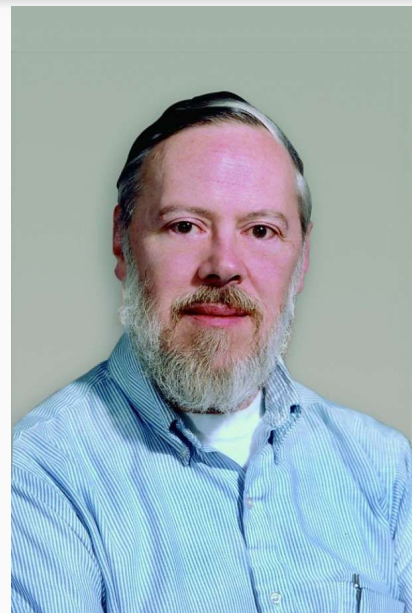
<https://www.youtube.com/watch?v=Za6vGTLp-wg>

# De MULTICS à UNIX : De 1964 à 1971



## Ken Thompson et Dennis Ritchie

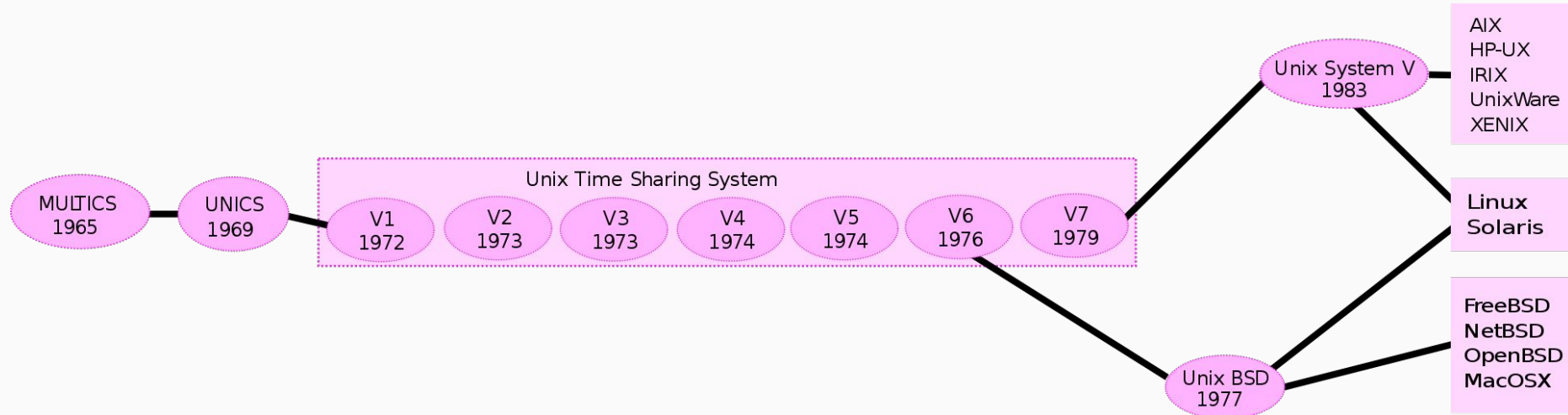
- MULTICS
- UNICS
- UNIX
- Langage C
- Code propriétaire AT&T Bell Labs
  - UNIX n'est **pas** libre



# À partir de 1971 : des versions d'UNIX et des dérivés

- UNIX évolue (branche de recherche AT&T)
- Une license UNIX qui coûte cher
- Des dérivés UNIX-based voient le jour
  - Famille **BSD** (Berkeley Software Distribution)
    - 1ere version OpenSource en Juin 1989 - Networking Release 1 (Net/1)
  - Famille **System V**
    - Version commerciale AT&T

# À partir de 1971 : des versions d'UNIX et des dérivés



# À partir de 1983

## Richard Stallman

- Fondateur de GNU (**GNU's Not UNIX**)
  - Déclencheur : Anecdote de l'imprimante
  - Objectif : Créer un **équivalent d'UNIX libre**
  - Fondateur de la **FSF** : Free Software Foundation
    - Créateur de la licence **GPL**
- Le noyau Hurd met du temps à être développé
- GNU de son côté avance bien
  - GCC : GNU Compiler Collection
  - GDB : GNU Debugger
  - Bash
  - glibc



# À partir de 1991

## Linus Torvalds

- Étudiant à l'Université de Helsinki
  - Utilisateur de MINIX créé par Andrew Stuart Tanenbaum
- Se lance dans le développement de son propre noyau
  - Pour le plaisir, indépendamment de tout OS dans le but d'utiliser les fonctions de son nouvel ordi 32 bits
  - MINIX suit un design 16 bits et est mal adapté aux fct. 32 bits
  - Développé sur MINIX, compilé avec GCC, inspiré de MINIX et libre de tout code propriétaire
- Linux v0.01 en 1991 sous licence GPL
- GNU/Linux : outils GNU, noyau Linux (remplace Hurd)



# À partir de 1991



Linus Benedict Torvalds



Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).

I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus ([torv...@kruuna.helsinki.fi](mailto:torv...@kruuna.helsinki.fi))

PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT protable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-).

Architecture  
32 bits

Intel 80386



Intel 80486



Intel Pentium



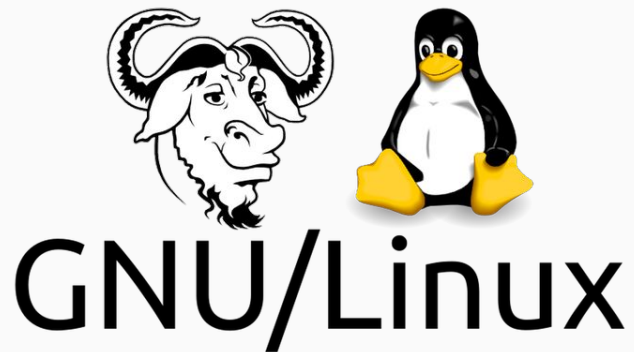


# Et ensuite

Une évolution en permanence

- Linux Kernel : Version 5.14.8 (26/09/2021)
- GNU/Linux : Des centaines de distributions

Norme POSIX : Définit les interfaces communes à tous les systèmes de type UNIX



# Rôles et contextes d'utilisation

# Rôles d'un système d'exploitation

Les rôles sont différents selon les cas d'utilisation :

- Télévision connectée
- Ordinateur
- Matériel spécifique
- Autres...

Selon vous, quels  
sont les rôles  
d'un OS ?

# Rôles d'un système d'exploitation

Un système d'exploitation remplit plusieurs fonctions :

- **(A) présenter** aux applications une interface dissimulant la complexité de l'infrastructure sous-jacente
- **(B) allouer** les ressources matérielles et logicielles pour satisfaire les besoins des applications
- **(C) contrôler** l'exécution des programmes pour éviter l'utilisation incorrecte des ressources

# Rôles d'un système d'exploitation

## Abstraction matérielle (A)

- Dissimuler la diversité et la complexité liée au matériel
- Fournir une interface unique de manipulation
  - Exemple : Stockage de masse
    - Constructeurs
    - Types (HDD, SSHD, SSD...)
    - Bus d'accès (IDE, SATA, PCI-E)
- Augmente la portabilité

# Rôles d'un système d'exploitation

## Gestion des ressources (B)

- Gère l'allocation du processeur
  - Ordonnancement
- Gère l'allocation mémoire
  - Mémoire physique
  - Mémoire virtuelle
- Gère les entrées/sorties
  - Gestionnaire de périphériques (Pilotes / Drivers)

# Rôles d'un système d'exploitation

## Assurer la sécurité (C)

- Contrôler l'accès aux ressources
  - Fichiers
  - Mémoire
- Conserver la bonne intégrité du système



# Rôles d'un système d'exploitation

## Résumé différemment

- Gestion du processeur (allocation, ordonnancement)
- Gestion de la mémoire (vive, virtuelle)
- Gestion des entrées/sorties (pilotes)
- Gestion de l'exécution des applications (processus)
- Gestion des droits (sécurité)
- Gestion des fichiers (file system)

# Contexte

La manière dont l'OS assure son rôle dépend du contexte :

- Par exemple, pour un ordinateur familial :
  - Facilité d'utilisation
  - Partagé entre plusieurs personnes
- Contraintes liées au domaine
  - Téléphone portable
  - Systèmes embarqués
  - Nécessité d'un certain temps de réponse (real time)

# Contexte

Divers contextes d'utilisation :



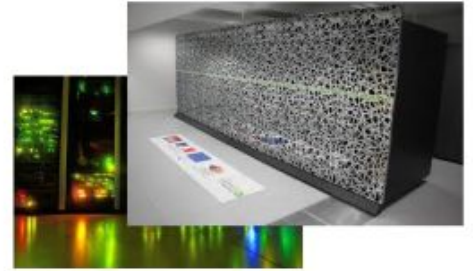
MAINFRAME



STATION DE TRAVAIL



SYSTÈMES EMBARQUÉS



SERVEURS - HPC

# Contexte

Les OS ont dû s'adapter aux évolutions du matériel et aux contextes d'utilisation :

- mono-utilisateur / multi-utilisateurs
- mono-tâche / multi-tâches
- mono-processeur / multi-processeur
- temps réel

# Contexte

Systèmes d'exploitation **multi-utilisateurs** :

- Partage des ressources de la machine
- Contrôle d'accès (sécurité)
- Notion de super-user (root)

# Contexte

Systèmes d'exploitation **multi-tâches** :

- Gestion de plusieurs processus “**simultanément**”
  - Ne signifie pas forcément que les tâches fonctionnent au même moment
    - Un seul coeur CPU (concurrency)
    - Plusieurs coeurs CPU (parallélisme)
  - Un processus ne doit pas accéder à la mémoire d'un autre processus
    - Cas particulier : mémoire partagée
- Préemptif ou collaboratif
  - Préemptif : L'OS garde la main sur les tâches exécutées
  - Collaboratif : Le processus décide quand il rend la main

# Structure d'un OS

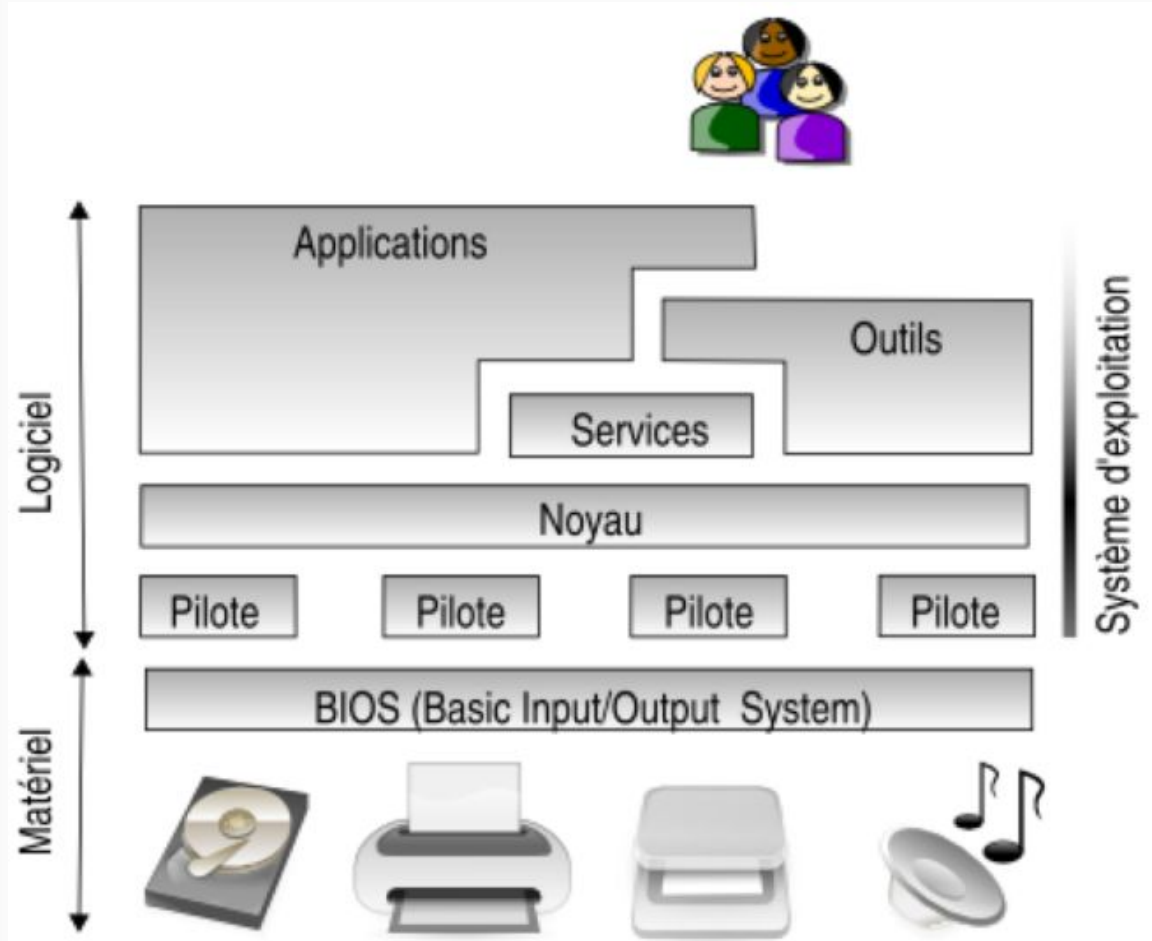
# Structure d'un Système d'exploitation

Il n'existe pas d'architecture standard pour la structure d'un système d'exploitation mais on observe toujours :

- Un noyau
  - Exemple : **Linux** dans **GNU/Linux**
  - Gère les ressources, les tâches, les évènements, les entrées/sorties...
- Un logiciel système
  - Exemple : **GNU** dans **GNU/Linux**
  - Rend le système d'exploitation utilisable
  - Fournit un environnement permettant d'exécuter des logiciels applicatifs



# Structure d'un Système d'exploitation

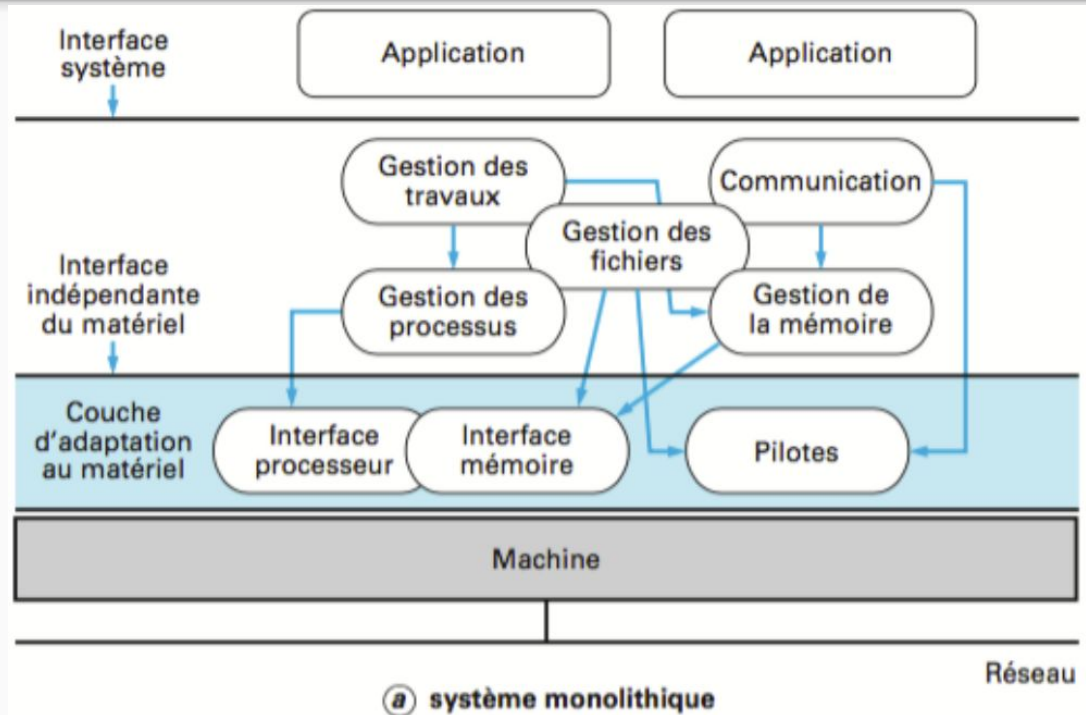


## Différents **types de noyau** :

- **Monolithique**
  - Tout dans un seul programme
  - Exemples : **Linux, DOS (<= Windows 98)**
- **Micro-noyau**
  - Le noyau n'effectue que les tâches primitives
  - Le noyau délègue à d'autres services
- **Hybride**
  - Micro-noyau mais qui délègue moins de choses
  - Exemples : **Windows NT, 2000 ... Windows 10**
- **Exo-noyau**
  - Imposer le moins d'abstraction matérielle possible

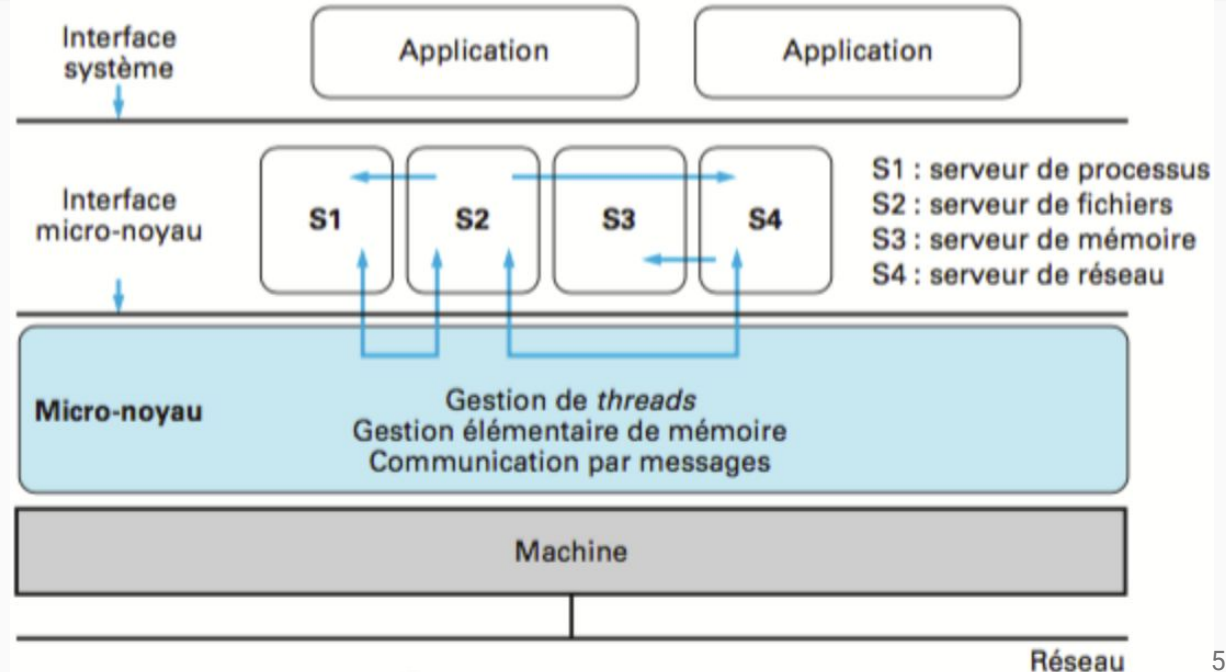
# Structure d'un Système d'exploitation

## Noyau **monolithique**



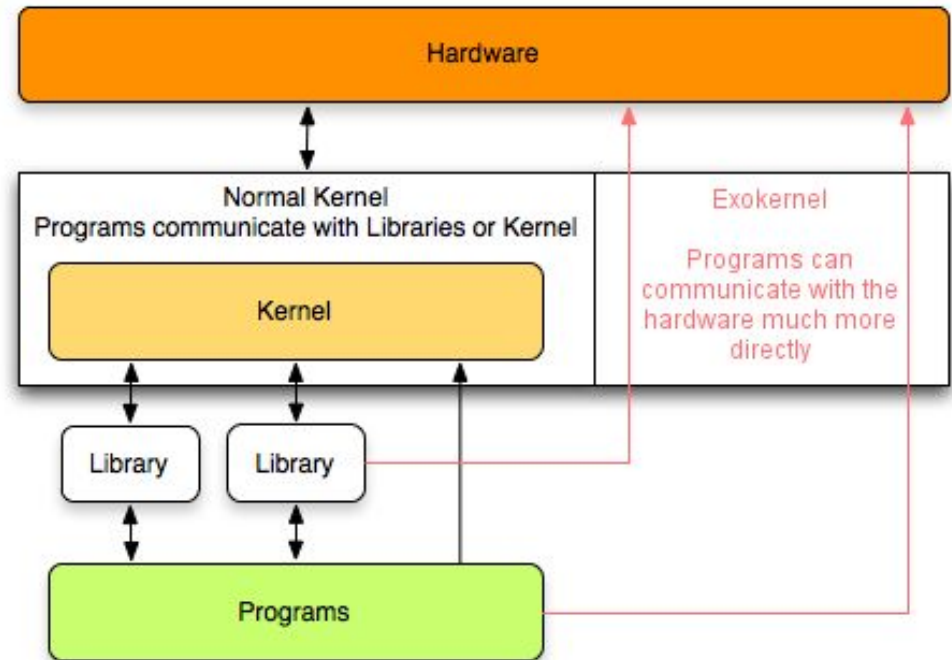
# Structure d'un Système d'exploitation

## Micro-noyau



# Structure d'un Système d'exploitation

## Exo-noyau



# Structure d'un Système d'exploitation

## Les **pilotes** (drivers)

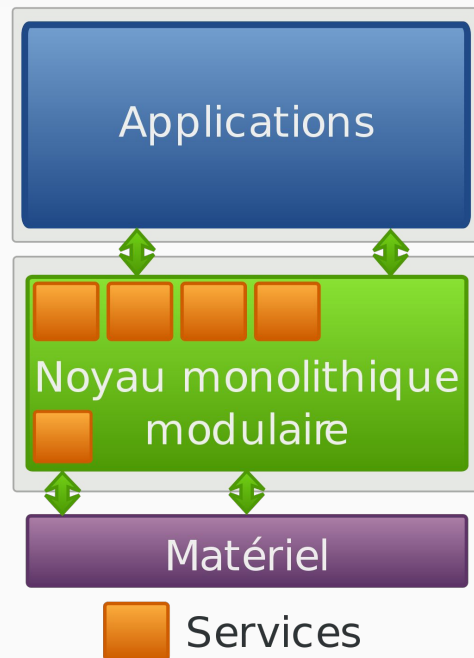
- Permettent d'interagir avec un périphérique
  - Clef USB, Dongle Wifi, manette de jeu, imprimante...
- Ils **font partie du noyau**
  - Le noyau est un programme compilé
    - Si on veut modifier le programme il faut recompiler
    - Il faut donc **recompiler à chaque nouveau pilote installé ??**

Une solution à ce  
problème ?

# Structure d'un Système d'exploitation

## La solution : les **modules**

- Chargés à la demande, remplaçables
- Séparés du binaire du noyau
- **Noyau Linux** : Depuis la version 2.0
  - Noyau **monolithique modulaire**
- Exemple: Commande **dkms**
  - Dynamic Kernel Module Support





Les interruptions,  
ça vous parle ?

# Structure d'un Système d'exploitation

## Les interruptions

- Interrompt l'exécution normale des instructions du programme pour en exécuter d'autres avant de revenir au fonctionnement normal
- **Événement** particulier qui peut être **logiciel** ou **matériel**
  - Exemples Hardware : Bouton reset, timer watchdog
  - Exemples Software
    - Appel système (exemple: demande d'accès à un fichier)
    - Erreurs (exemple: division par 0)

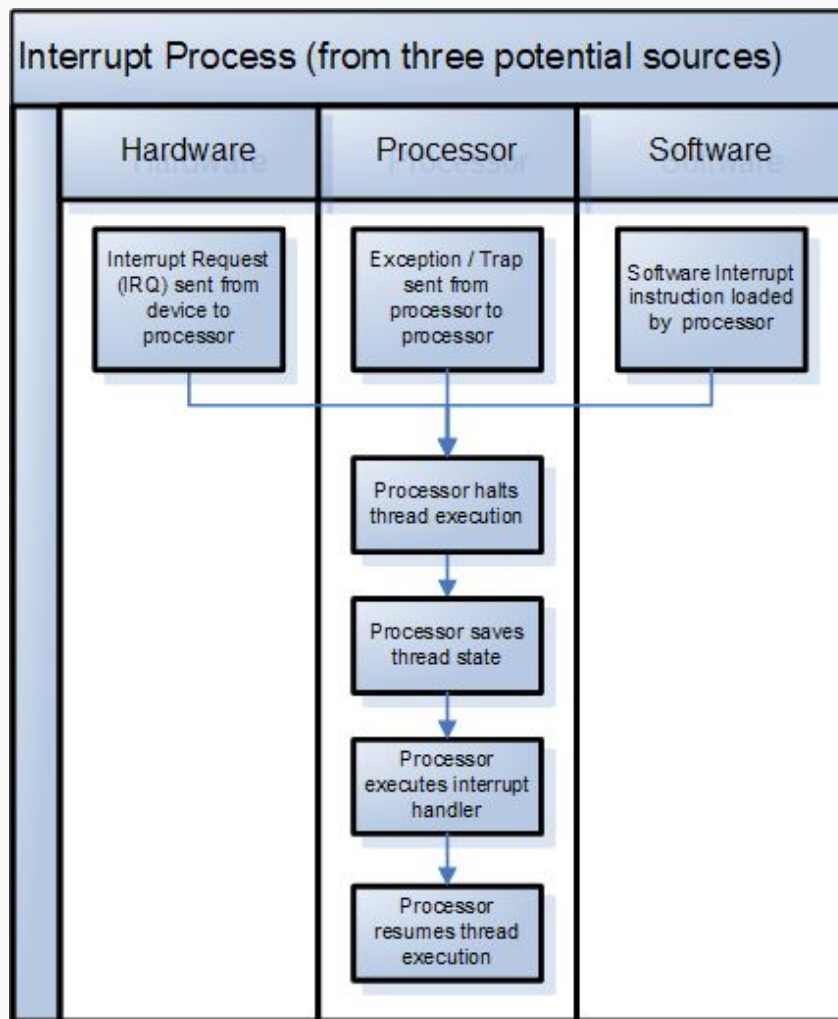
# Structure d'un Système d'exploitation

Lorsqu'une **interruption** se produit

- Le processeur est alerté d'une interruption
- La requête d'interruption est acceptée
  - Le processeur suspend l'activité en cours
  - Sauve l'état de cette activité (registres CPU)
  - Exécute la routine liée à l'interruption (**ISR**: Interrupt Service Routine)
- Reprend sa précédente activité
  - Restaure l'état de l'activité

# Structure d'un Système d'exploitation

## Interrupts

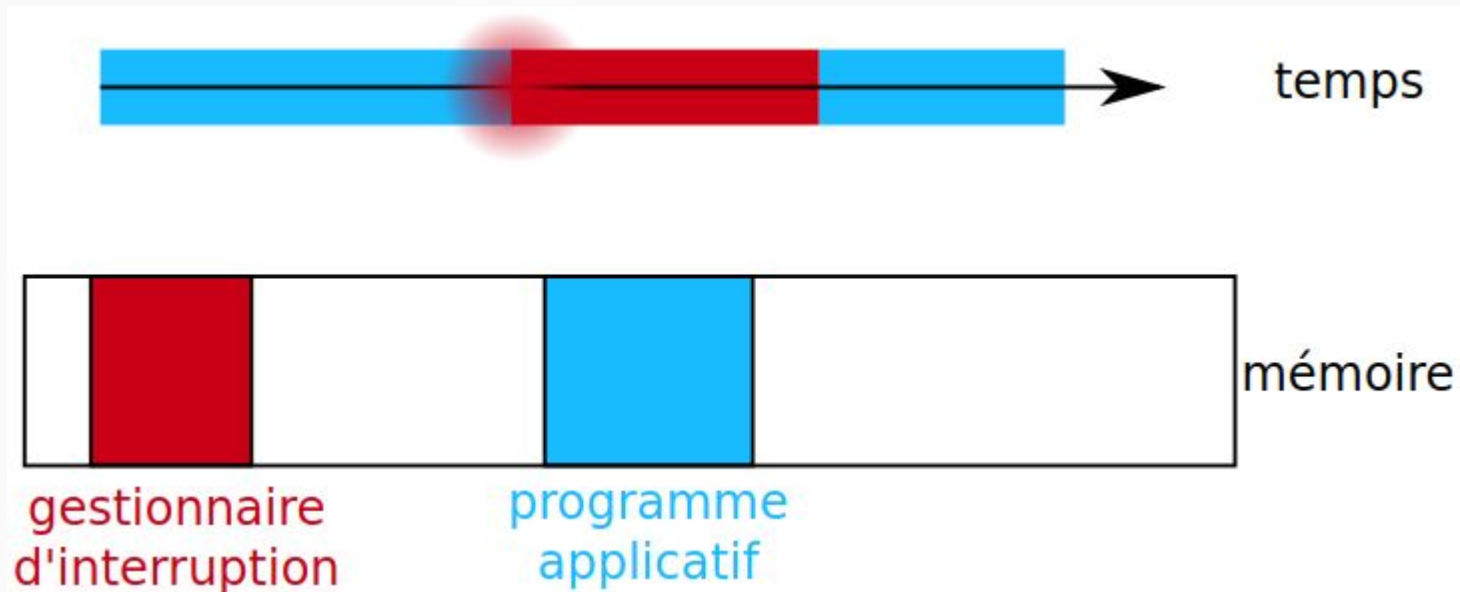


# Structure d'un Système d'exploitation

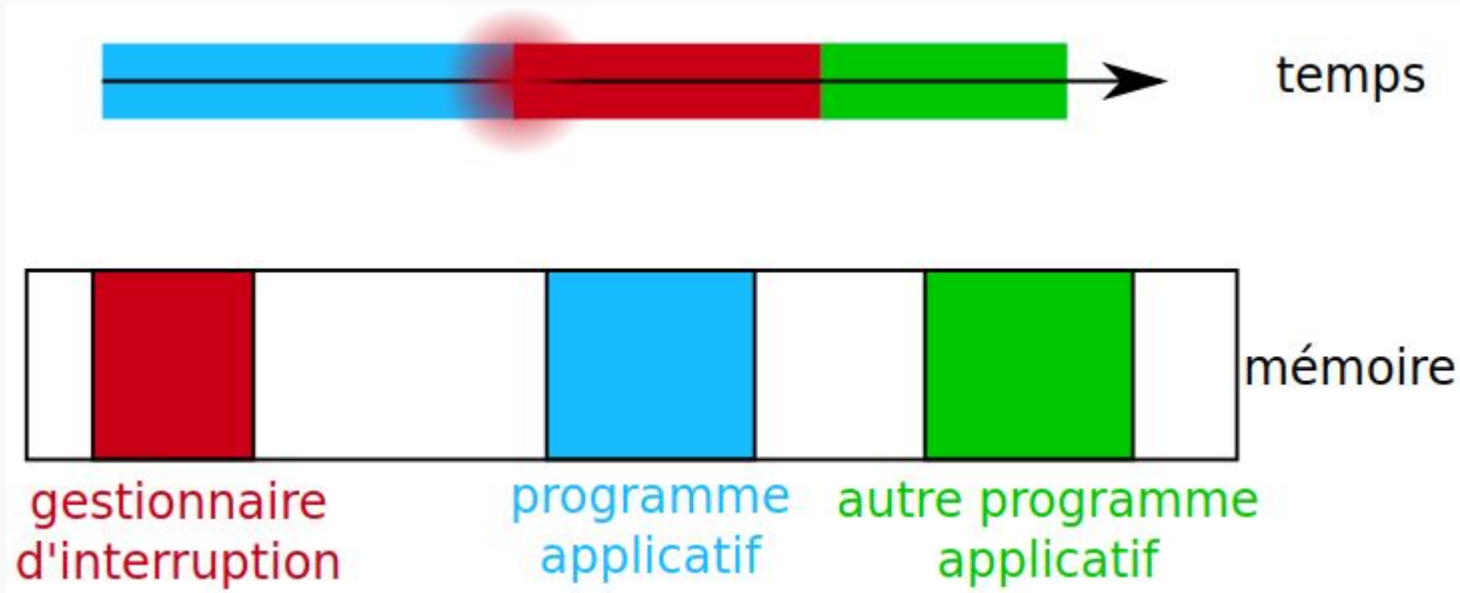
Une **interruption** permet notamment

- De gérer des communications non bloquantes
- D'effectuer du multitâches, c'est souvent la base d'OS **multitâches**
  - Alerter le système d'un événement
- Économiser de l'énergie
  - Le CPU n'a pas besoin de vérifier en boucle quelque chose, une interruption peut venir le "réveiller"
  - Moins d'émission de chaleur
  - Très utile pour de l'embarqué qui fonctionne sur batterie

# Structure d'un Système d'exploitation



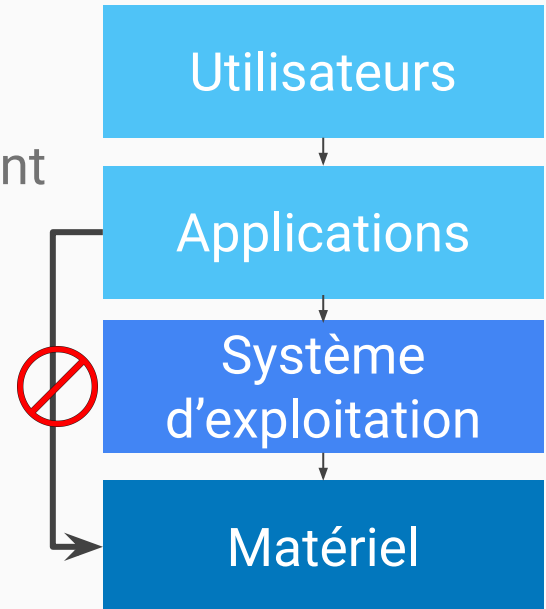
# Structure d'un Système d'exploitation



# Structure d'un Système d'exploitation

## Autorisation et arbitrage

- Le noyau doit être le seul à accéder directement aux ressources
  - Gestion des autorisations
  - Gestion de l'arbitrage





# Structure d'un Système d'exploitation

## Autorisation et arbitrage

- Il faut vérifier qu'une ressource n'est pas déjà **en cours d'utilisation**
- Il faut vérifier que le demandeur **est autorisé** à y accéder

Le système possède des mécanismes pour gérer tout cela

- Mets à disposition des **appels systèmes** de différents types
- Plusieurs gestionnaires d'interruptions

# Structure d'un Système d'exploitation

## Appels systèmes

- Un intermédiaire permettant d'accéder au matériel
- Le noyau peut gérer les appels qu'il reçoit, **autoriser** et **arbitrer**
- Les applications peuvent demander des services fournis par le noyau
  - Allocation mémoire
  - Lecture / Écriture de fichier

# Structure d'un Système d'exploitation

Il existe de nombreux **types d'appels systèmes**

- Gestion de **processus**
  - Tuer un processus, allouer de la mémoire à un programme...
- Manipulation de **fichiers**
  - Lire, écrire...
- Utilisation de **périphériques**
  - Éjecter un volume de stockage
- **Communication, protection, information...**

# Structure d'un Système d'exploitation

## Modes d'exécution

- **Utilisateur**
  - Certaines instructions sont limitées ou interdites
  - Les applications s'exécutent dans ce mode
- **Superviseur** (aussi appelé **mode noyau**)
  - Accès à toutes les instructions CPU
  - Les **appels systèmes** permettent aux applications de demander **au noyau** d'exécuter des opérations en **mode superviseur**
  - Les gestionnaires **d'interruptions** utilisent le **mode superviseur**

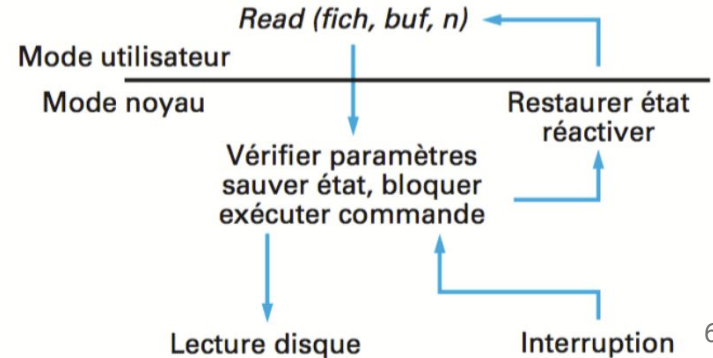
# Structure d'un Système d'exploitation

## Mode d'exécution et appel système

- Appel effectué à travers une **API** fournie par l'OS

```
#include <unistd.h>
```

```
ssize_t read(int fd, void* buf, size_t n);
```



# Structure d'un Système d'exploitation

**Exemples d'API**  
permettant  
d'effectuer des  
**appels systèmes**

Appel	Windows	Linux
Gestion processus	CreateProcess()	fork()
	ExitProcess()	exit()
	WaitForSingleObject()	wait()
Gestion de fichier	CreateFile()	open()
	ReadFile()	read()
	WriteFile()	write()
Protection	SetFileSecurity()	pipe()
	CreateFileMapping()	umask()
	SetSecurityDescriptorMask()	chown()

# Structure d'un Système d'exploitation

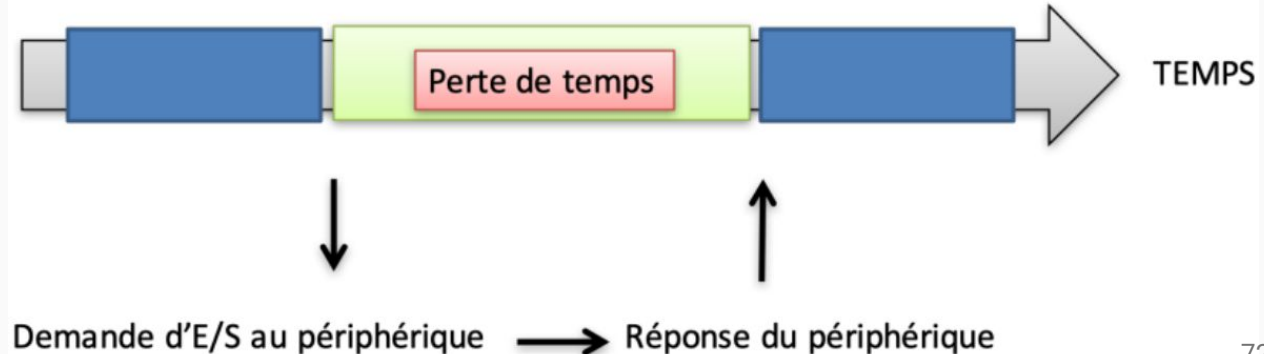
## Entrées / sorties

- Très **lent** comparé au processeur
  - Exemple : Disque dur HDD
- Perte de temps
  - On peut optimiser grâce aux **interruptions**

# Structure d'un Système d'exploitation

## Entrées / sorties

- On attend
  - Perte de temps

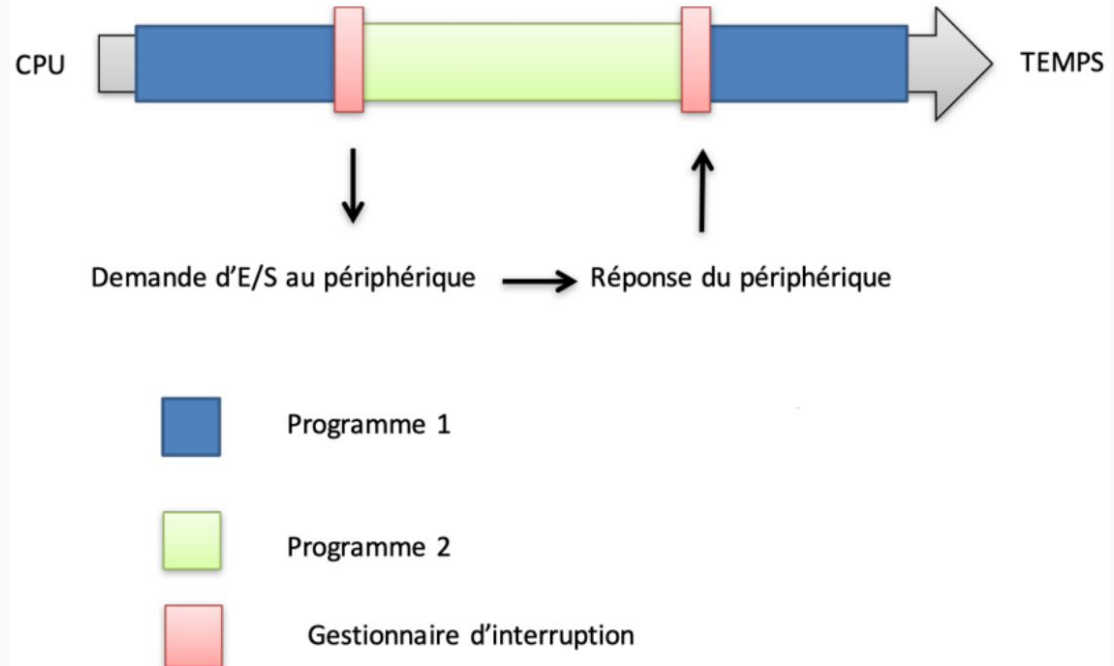




# Structure d'un Système d'exploitation

## Entrées / sorties

- On donne la main
  - Optimisation



# Structure d'un Système d'exploitation

## Ordonnanceur (scheduler)

- **Arbitre** les programmes qui s'exécutent
- Gère le CPU
  - Détermine le programme qui prend la main suite à une **interruption**
- Implémentation faite de compromis
  - Doit assurer l'équité entre les **processus**
  - Éviter les problèmes d'**attente** trop longue

# Structure d'un Système d'exploitation

## Ordonnanceur **collaboratif** :

- Plus simple que le préemptif
- Les tâches **rendent la main** quand elles sont finies ou quand elles le décident
- Système plus instable (boucle infinie)

## Ordonnanceur **préemptif** :

- Le **système décide** quand mettre en pause une tâche
- Système plus stable
  - Une erreur dans un programme ne compromet pas tout le système
  - Une boucle infinie ne bloque pas tout le système

# Structure d'un Système d'exploitation

## Ordonnanceur préemptif : Définition Wikipédia

**Preemption** is the act of temporarily interrupting a task being carried out by a computer system, without requiring its **cooperation**, and with the intention of resuming the task at a later time.

La **préemption** est le fait de pouvoir interrompre une tâche sans nécessiter sa **coopération**, avec l'intention de la continuer plus tard.

- Rappel : Système d'Exploitation **multitâches** préemptif

# Structure d'un Système d'exploitation

Les Systèmes d'Exploitation **modernes** utilisent un ordonnanceur **préemptif**.

- Collaboratif
  - MS-DOS, Windows 3.1
- Préemptif
  - Windows NT à Windows 10
  - MacOS
  - UNIX
  - Linux

# Structure d'un Système d'exploitation

## Ordonnanceur préemptif : Le principe

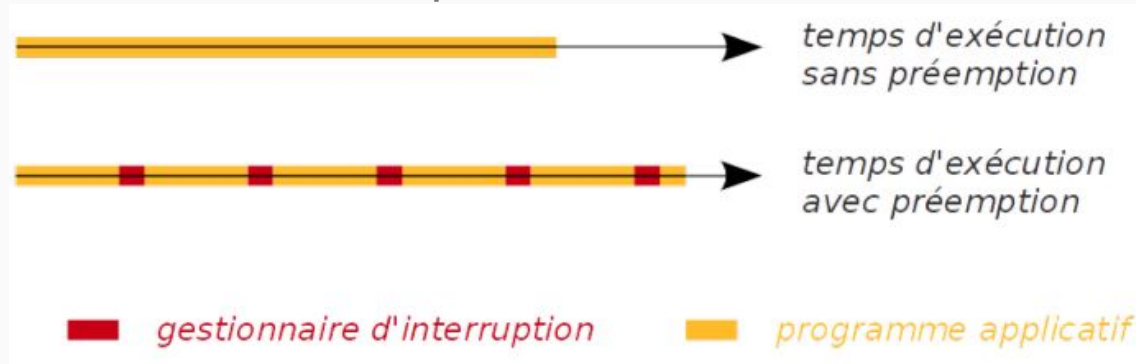
An **interrupt** is scheduled to allow the **operating system kernel** to switch between **processes** when **their time slices expire**, effectively allowing the processor's time to be shared between a number of tasks, giving **the illusion** that it is dealing with these tasks in **parallel** (simultaneously). The operating system which controls such a design is called a **multi-tasking system**.

Une **interruption** programmée permet au noyau de changer de **processus**, ce qui permet de partager le temps CPU entre plusieurs tâches, donnant **l'illusion** que le système gère ces tâches en **parallèle**. Ce genre d'OS est appelé système **multitâches**.<sup>78</sup>

# Structure d'un Système d'exploitation

## Ordonnanceur préemptif : l'Horloge (timer programmable)

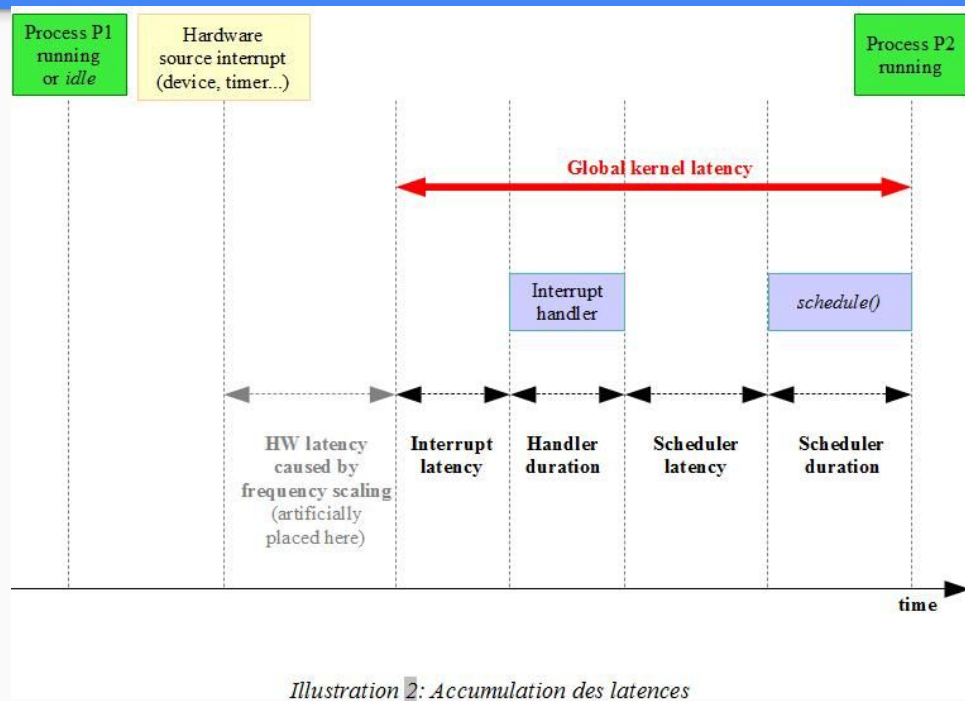
- Déclenche des **interruptions** à intervalle régulier
- Garantit un **arbitrage** régulier par **préemption** (**multitâche préemptif**)
- Provoque un surcoût en temps de calcul



# Structure d'un Système d'exploitation

## Latence

- Les interruptions induisent une latence
- L'horloge ne doit pas être trop rapide ni trop lente
  - Imprécision
  - Overhead





# Structure d'un Système d'exploitation

Pour s'orienter vers un **système temps réel** :

- Exemple : Ce que fait **Linux-rt**
  - Rendre préemptible la majeure partie du noyau
  - Utiliser une **horloge plus rapide**
    - Présence de mécanismes pour réduire les temps de latence induits
  - <https://fr.wikipedia.org/wiki/Linux-rt>

# Amorçage

# Amorçage

Au démarrage d'un ordinateur :

- Le **firmware** est chargé en mémoire
- La **zone d'amorçage** est scannée sur chaque **périphérique de stockage**
- Le **code d'amorçage** s'exécute et **charge le noyau en mémoire**

# Amorçage

## Le firmware :

- Le **micrologiciel** de la carte mère
- Initialement **BIOS**, remplacé par l'**UEFI**
- Effectue des tests du matériel (alim OK, présence d'un clavier...)
- Recherche les **périphériques de stockage** existants
- Généralement stocké en **EEPROM** dans la carte mère
  - **Electrically-Erasable Programmable Read-Only Memory**
    - Mémoire morte reprogrammable
  - Permet le flash du **BIOS**

Bonus : Comment  
sont conservés les  
paramètres du BIOS ?

# Amorçage

## (Bonus) CMOS RAM

- **Mémoire volatile** faiblement alimentée
- Réinitialisation des paramètres du **BIOS** possible en cas de plantage



# Amorçage

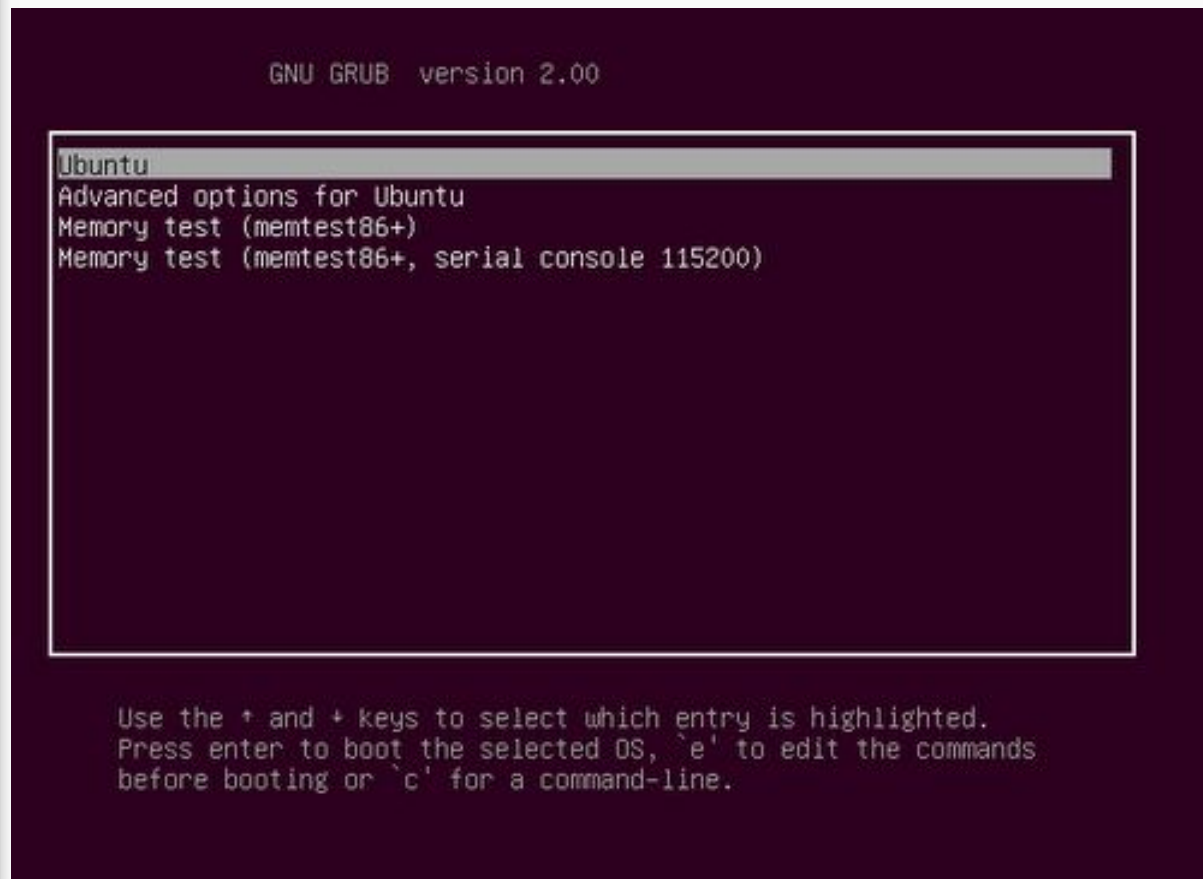
## La zone d'amorçage :

- Permet de charger le **code d'amorçage** ou **boot loader**
- Scannée sur chaque **périphérique de stockage** (configurable)
- Contient la **table de partitionnement** du périphérique de stockage
  - Le **secteur d'amorçage principal** est limité en mémoire
    - Le code d'amorçage initial peut appeler un autre **bootloader** (ex: **GRUB**)
    - Les **bootloaders** peuvent être chaînés (**GRUB -> Windows Loader**)
    - Ils peuvent être stockés à différents endroits
  - De type **MBR** (époque du **BIOS**) : Master Boot Record
  - De type **GPT** (époque de **l'UEFI**) : GUID Partition Table

# Amorçage

Exemple d'interface du  
**bootloader GNU GRUB** en  
version 2.

Ce **bootloader** a la capacité  
de pouvoir lancer au choix un  
système d'exploitation parmi  
**plusieurs OS** disponibles sur  
la machine.





Connaissez-vous  
les différences  
entre MBR et GPT ?

# Amorçage

## Master Boot Record (MBR) :

- Jusqu'à **4** partitions primaires
- Ne supporte pas les disques dur de plus de **2 TB** ( $2^{32} * 512$ )
  - Les **tailles de partitions** sont stockées sur 4 octets (**32 bits**)
  - Les **secteurs** sont limités à une taille de **512 octets**

## GUID Partition Table (GPT) :

- Jusqu'à **128** partitions
- Supporte théoriquement des disques jusqu'à **9.4 ZB**
  - Les **tailles de partitions** sont stockées sur 8 octets (**64 bits**)

Code d'amorçage  
exécuté et noyau  
chargé, ensuite ?

# Amorçage

Ensuite, le noyau et l'OS remplissent leurs rôles

- Exécute des processus
- Gère des interruptions
- ...

En gros, tout ce qu'on a dit dans le cours depuis le début et certainement plus encore !

# Installation d'un OS

D'après vous  
comment ça  
s'installe ?

# Installation d'un OS

En gros, mêmes principes que l'amorçage mais cette fois on amorce un **programme d'installation** ou une version "**Live CD**" du système.

Différents supports de stockage

- **Amorçage** via Clef USB, CD, Disquette, DVD...

Installation via le réseau

- **Amorçage PXE**
  - Preboot Execution Environment

- Cours de Romain Franceschini des années précédentes
- Cours de Pierre-Antoine Champin :  
<https://perso.liris.cnrs.fr/pierre-antoine.champin/enseignement/se/intro.html>
- « Systèmes d'exploitation : principes et fonctions » S. Krakowiak, Techniques de l'Ingénieur
- <https://fr.statista.com/statistiques/559498/parts-de-marche-des-systemes-d-exploitation-pour-ordinateurs-de-bureau-console-tablette/>
- <https://labo.fnac.com/actualite/windows-10-passe-barre-50-parts-marche/#:~:text=Au%20total%2C%20Windows%20domine%20donc,encore%20%C3%A0%201%2C57%20%25>
- <https://linux.developpez.com/actu/306993/Linux-continue-sa-croissance-et-passe-a-3-61-pourcent-de-part-sur-le-marche-des-OS-desktop-d-apres-NetMarketShare-sa-plus-haute-part-de-marche-jamais-atteinte-sur-ce-barometre/#:~:text=Windows%20a%20enregistr%C3%A9%20une%20part,8.1%20avec%203%2C25%20%25>
- [https://fr.wikipedia.org/wiki/Microsoft\\_Windows](https://fr.wikipedia.org/wiki/Microsoft_Windows)
- <https://en.wikipedia.org/wiki/MacOS>
- [https://en.wikipedia.org/wiki/Intel\\_80386](https://en.wikipedia.org/wiki/Intel_80386)
- [https://fr.wikipedia.org/wiki/Multit%C3%A2che\\_pr%C3%A9emptif](https://fr.wikipedia.org/wiki/Multit%C3%A2che_pr%C3%A9emptif)
- <https://en.wikipedia.org/wiki/Exokernel>
- <http://spiralconnect.univ-lyon1.fr/spiral-files/download?mode=inline&data=2189794>
- [https://dchabal.developpez.com/tutoriels/linux/xenomai/?page=page\\_3](https://dchabal.developpez.com/tutoriels/linux/xenomai/?page=page_3)
- <https://www.mustbegeek.com/difference-between-mbr-and-gpt/>
-