Chapitre 1

Le choix Linux

ENSI CAEN

ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

2021-2022

SYSTÈMES D'EXPLOITATION

Historique des systèmes d'exploitation

Noyau d'un OS

ENSI CAEN

ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

Définition

Un **système d'exploitation** ou ***Operating System* (OS)** est un programme qui pilote un système informatique et électronique en contrôlant ses ressources.

Cela inclut :

- Gestion des fonctions périphériques matérielles (drivers ou pilotes)
- Gestion virtuelle de la mémoire vive principale (RAM) par gestion de l'unité de contrôle de la mémoire (MMU)
- Gestion de la mémoire de masse (HDD, SSD, SD, etc) et des fichiers dans une arborescence (*File System*) : organisation, implantation
- Gestion des processus et threads (gestion du/des CPU) : création, ordonnancement, planification, coopération, dépendance, …
- Gestion de la sécurité et des politiques d'accès : multi-utilisateurs, multi-tâches
- etc

En revanche, un OS n'est pas :

- Un logiciel applicatif (traitement de texte, compilateur, navigateur Web, …)
- Une interface graphique (GUI, CLI)

Définition

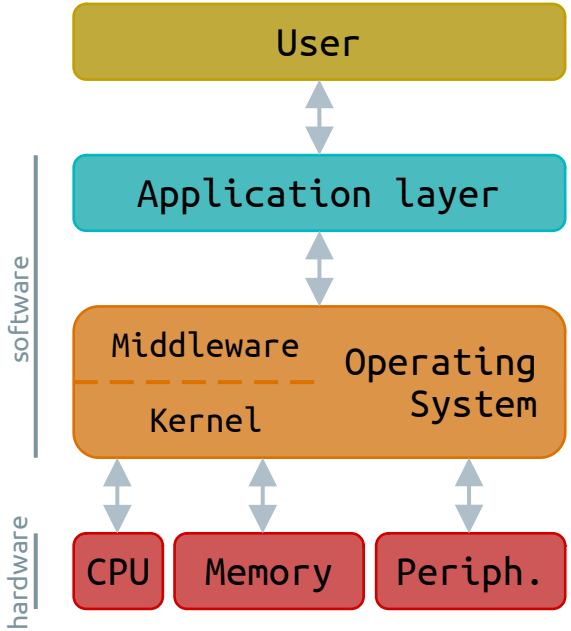
On peut considérer l'OS comme étant un intermédiaire entre les applications logicielles et les ressources matérielles de la machine sur laquelle il fonctionne.

Un OS est un gestionnaire de ressources proposant des services logiciels accessibles depuis les applications :

Gestion des ressources matérielles (CPU, mémoire de masse ou de travail, périphériques, …) pour les attribuer de manière optimale entre les différents processus qui les demandent.

Il peut aussi être vu comme une couche d'abstraction :

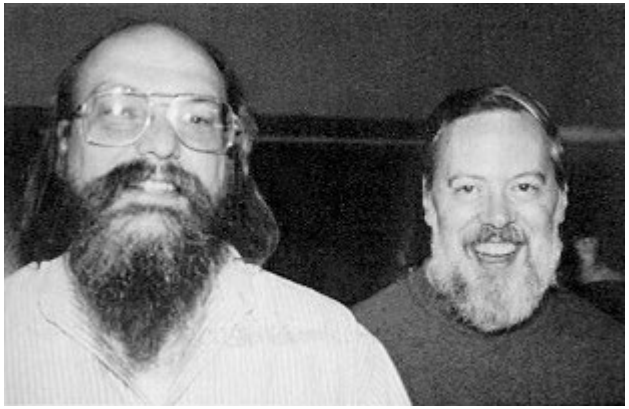
Il masque les mécanismes de fonctionnement de la machine dans le but de présenter une interface simple au développeur.



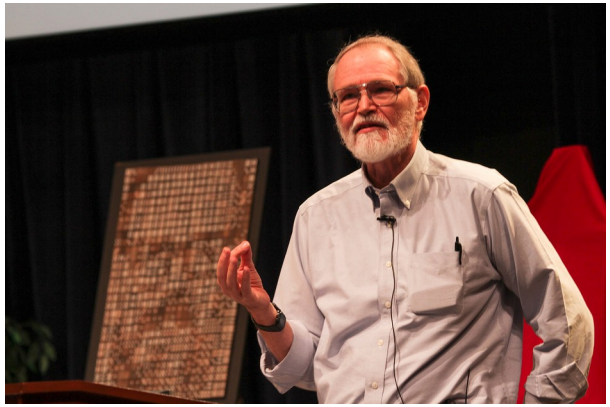
ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

Premiers OS

- Années 40 et 50 : pas d'OS, les systèmes sont mono-tâche et mono-utilisateur
- Années 60 : le MIT crée le premier OS, Multics. Il est multi-programmes, multi-utilisateurs
- Années 70 : UNIX développé par Thompson et Ritchie (Bell Labs), OS pour lequel le C est créé (Kernighan et Ritchie)



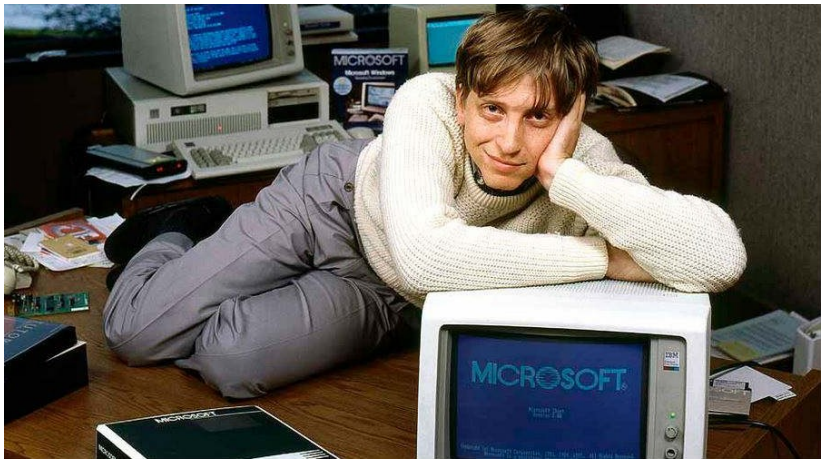
Ken Thompson et Dennis Ritchie.



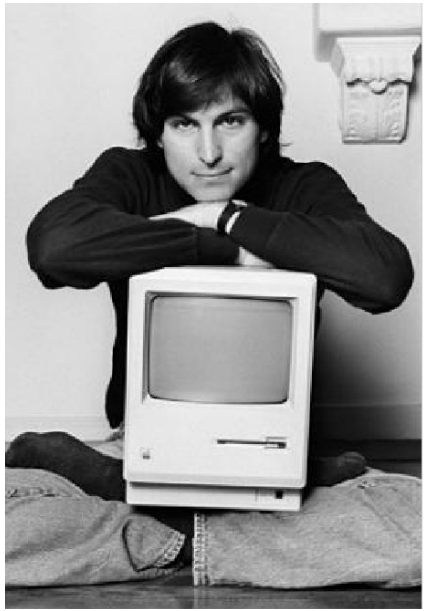
Brian Kernighan,
Hommage à D. Ritchie en 2012 (Bell Labs).

OS propriétaires

- Années 80 : IBM et Microsoft fondent le MS-DOS
- En parallèle : Xerox et Steve Jobs développent Xerox Star, abandonné puis réadapté pour Macintosh



Bill Gates

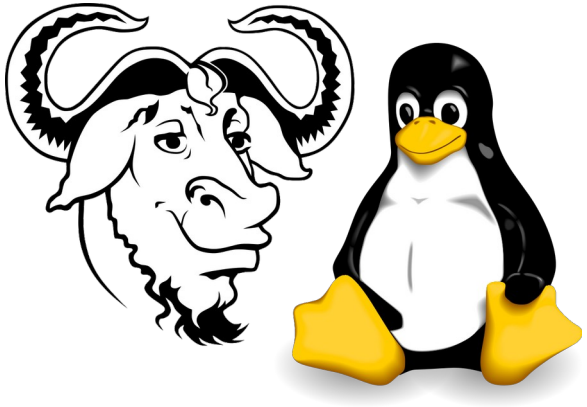


Steve Jobs

OS GNU/Linux

- 1983 : Stallman (MIT) crée GNU. 1er sous licence libre (la GNU GPL) mais c'est un OS sans noyau
- 1991 : Torvalds (Univ. d'Helsinki) développe le noyau Linux (licence GPL)
- 1994 : GNU et Linux s'associent pour donner naissance au premier OS 100 % libre : GNU/Linux

Richard Matthew Stallman



Logos GNU et Linux

Linus Torvalds

ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

Évolutions technologiques

Années 1960

Première génération : Système de traitement par lots

Exécution de grands calculs successifs, peu d'intervention utilisateur

Années 1970

Deuxième génération : Systèmes multi-programmés

Exécution des programmes par *scheduling* (ordonnancement ou planification)

Troisième génération : Systèmes en temps partagé

Années 1980

Le *scheduling* cherche à répondre aux demandes de plusieurs utilisateurs en communication directe

Années 2000

Quatrième génération : Systèmes temps-réel

L'objectif est de garantir l'exécution des tâches en un temps donné

Années 2010

Cinquième génération : Système distribué

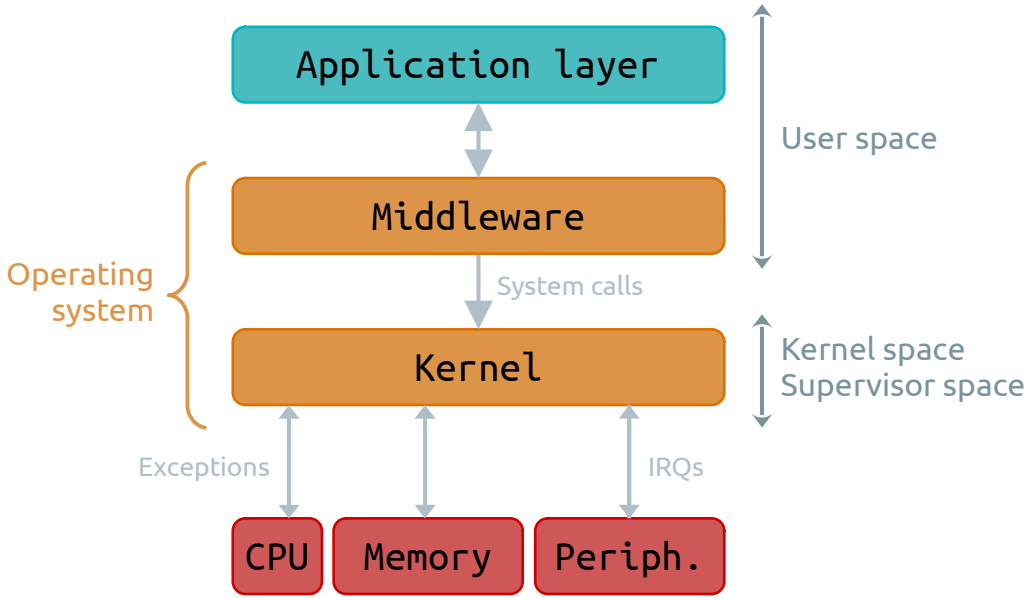
Gestion des ressources de plusieurs ordinateurs en simultané, via réseau informatique

Ce groupe de machine est vu comme une unique machine virtuelle, aux capacités importantes

Le kernel (noyau)

Dans un OS, Le ***kernel*** est l'interface la plus basse du modèle en couches logicielles de l'application.

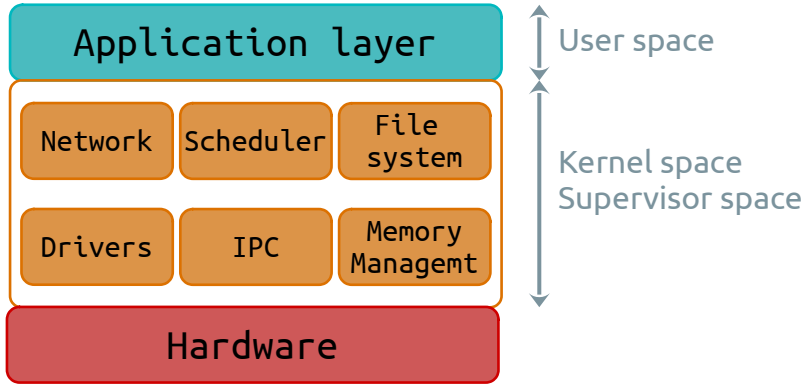
Il propose des services de bas niveaux (*scheduler, drivers*, *file system*, *network*, *memory management*, …) et son espace d'adressage est souvent virtualisé et associé à la notion de niveaux de privilège processeur (mécanisme de protection mémoire).



Il existe différentes familles de *kernels*. Découvrons les 4 principales.

Monolithic kernel

Tous les services bas niveaux évoluent dans le même espace d'adressage et sont encapsulés dans un binaire unique. Ceci implique une forte dépendance des outils systèmes entre eux (un bug dans un driver peu faire tomber le système complet) et une grande difficulté à maintenir de gros systèmes (ex. : GNU/Linux < 1.2 …).

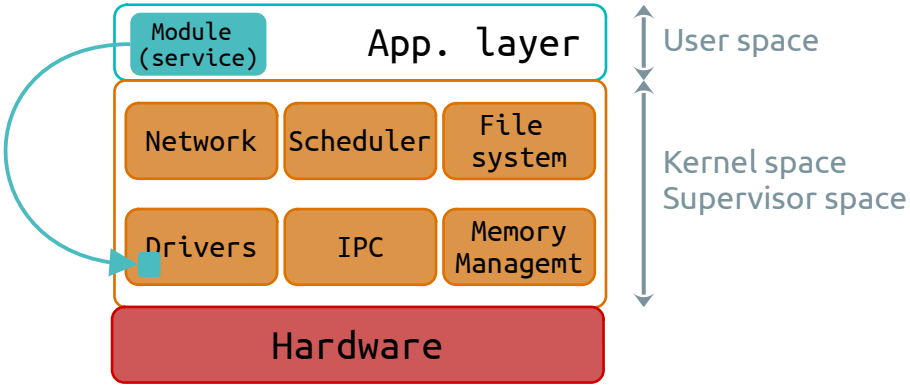


Monolithic modular kernel

Il s'agit de noyaux monolithiques avec une approche modulaire dynamique, impliquant le chargement à chaud de modules *kernel* (*runtime*).

Cette solution permet de n'inclure que les services nécessaires dans l'espace *kernel* puis d'en rajouter à chaud en fonction des besoins (solution modulable très pratique pour des phases de prototypage de drivers).

Quelques exemples : GNU/Linux > 1.2, FreeBSD, Solaris …

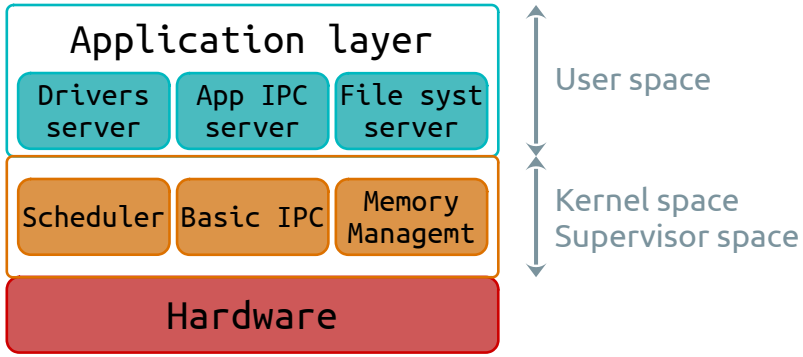


Microkernel

Le kernel n'offre **que les services les plus élémentaires et critiques** dans son espace d'adressage non protégé. Les autres services sont alors répartis dans des serveurs en espace utilisateur possédant leurs propres espaces d'adressages (confinement des défauts).

Il s'agit de solution plus fiable et robuste et surtout plus simple à maintenir … mais moins performantes (énormément d'appels systèmes).

Exemples de systèmes : MINIX, L4, SPARTAN …



Hybrid kernel

Le grand manque de performance des microkernels à amené la création des noyaux hybrides. Il s'agit de technologies mixtes entre les solutions monolithiques et les microkernels.

Nous pouvons retrouver dans l'espace noyau des services non critiques mais générateurs de grand nombre d'appels système.

Nous retrouvons notamment dans cette famille les OS suivants :

Windows XP, Vista, 7, 8, tous basés sur le noyau hybride NT (New Technology, ex : NT6.3 pour windows 8.1) ;

Mac OS X basé sur le noyau hybride XNU (X is Not Unix) dérivé de Mach kernel (microkernel) et de FreeBSD kernel (monolithic kernel).

LINUX EN BREF

De UNICS à Linux

Le kernel Linux

GNU, distributions GNU/Linux

ENSI CAEN

ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

UNIX (historiquement nommé UNICS) est un système d'exploitation multi-tâches, multi-utilisateurs (cloisonnement des espaces utilisateurs, *root* a tous les droits) créé en 1969 et basé sur un kernel monolithique.

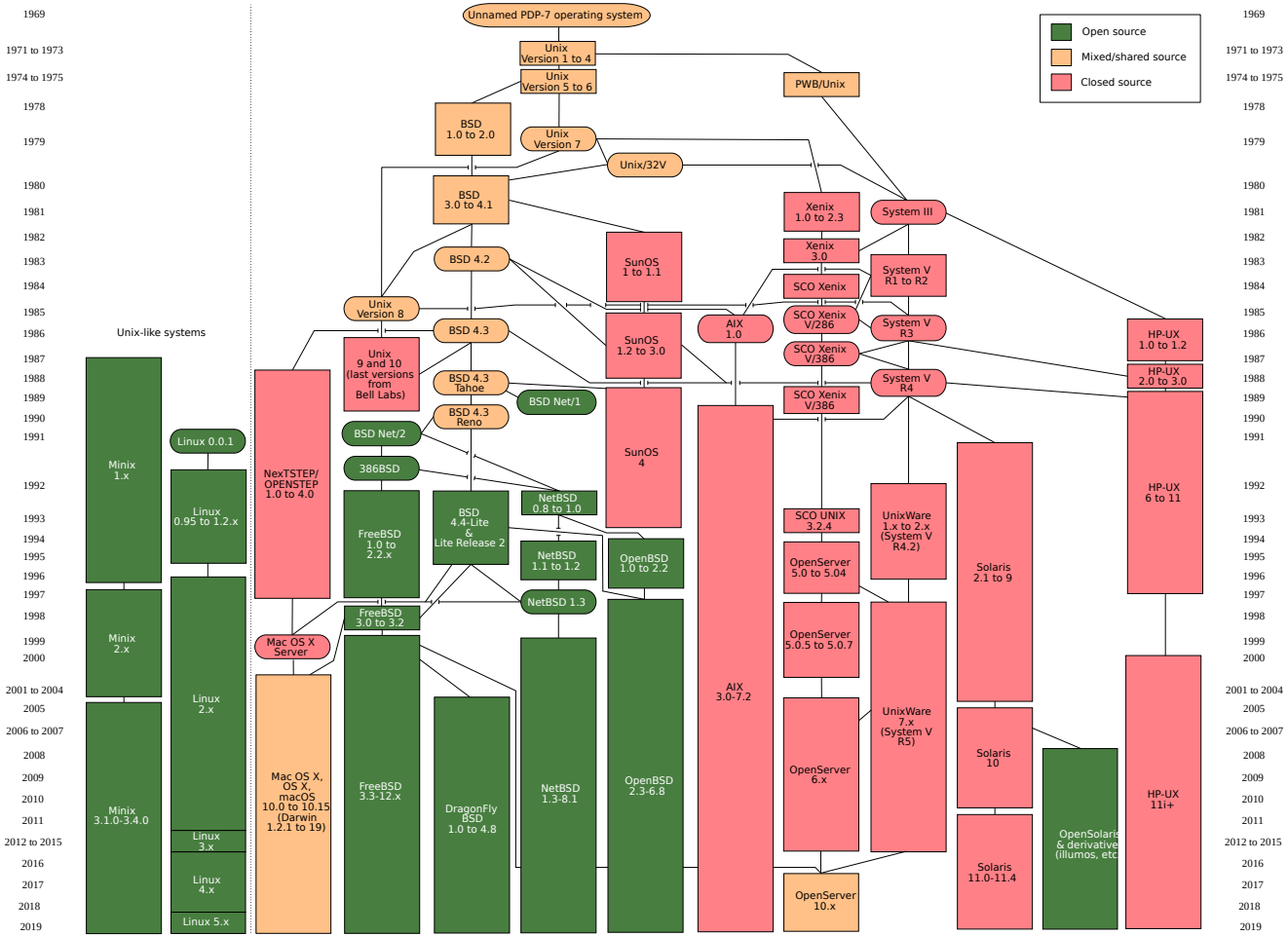
UNIX est à la base d'un (très) grand nombre de systèmes : nous parlons souvent de famille UNIX ou UNIX-like (GNU/Linux, FreeBSD, Mac OS X, Minix, Solaris …).

Dans cette famille, nous pouvons rencontrer aussi bien des solutions propriétaires que des solutions libres et OpenSource soumises à différents types de licences (GPL, LGPL, Apache, BSD …).

Vous aurez à suivre une conférence propre au monde de l'OpenSource et aux règles de déploiement de solutions logicielles mixtes (propriétaires/libres).

LINUX EN BREF

D’UNICS à Linux



Source :
https://fr.wikipedia.org/wiki/Unix

Le kernel Linux est une implémentation libre d'UNIX respectant les spécifications POSIX (norme IEEE 1003).

Il a été créé par Linus Torvalds et est né en réponse à Andrew Tanenbaum (créateur microkernel MINIX) qui ne souhaitait pas intégrer des contributions visant à améliorer MINIX.

Linux a été créé *from scratch* par Torvalds sur un modèle collaboratif décentralisé via Internet et réutilise les composants logiciel du projet GNU.

Pour info, Torvalds se mit à utiliser un logiciel de gestion de version en 2002 seulement. En 2005, il crée Git notamment pour correspondre à la philosophie libre de Linux.

WHAT IS LINUX?

Linux is a clone of the operating system Unix, written from scratch by Linus Torvalds with assistance from a loosely-knit team of hackers across the Net. It aims towards POSIX and Single UNIX Specification compliance.

It has all the features you would expect in a modern fully-fledged Unix, including true multitasking, virtual memory, shared libraries, demand loading, shared copy-on-write executables, proper memory management, and multistack networking including IPv4 and IPv6.

ON WHAT HARDWARE DOES IT RUN?

Although originally developed first for 32-bit x86-based PCs (386 or higher), today Linux also runs on (at least) the Compaq Alpha AXP, Sun SPARC and UltraSPARC, Motorola 68000, PowerPC, PowerPC64, ARM, Hitachi SuperH, Cell, IBM S/390, MIPS, HP PA-RISC, Intel IA-64, DEC VAX, AMD x86-64, AXIS CRIS, Xtensa, Tilera TILE, AVR32, ARC and Renesas M32R architectures.

Linux is easily portable to most general-purpose 32- or 64-bit architectures as long as they have a paged memory management unit (PMMU) and a port of the GNU C compiler (gcc) (part of The GNU Compiler Collection, GCC). Linux has also been ported to a number of architectures without a PMMU, although functionality is then obviously somewhat limited.

https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/admin-guide/README.rst?h=v6.0-rc3

Linux est probablement le noyau supportant en 2022 le plus d'architectures matérielles de CPU (tout type de kernels et de domaines confondus).

À titre indicatif, en 2020 le kernel représente environ 27.800.000 (66 492 fichiers et plus de 21000 auteurs) de lignes de code contre 176.250 pour la *release* 1.0.

https://www.kernel.org/

Le modèle de développement, de supervision et de validation des *releases* reste très hiérarchisé. Linus Torvalds supervise les changements de code et des *releases* des dernières versions, pilote toujours une grande partie des commits (28.815 commits soit 6.775.000 de lignes "pushées" à la lui seul) mais délègue néanmoins à d'autres développeurs experts le suivi d'une grande partie du noyau ainsi que la maintenance des *releases* antérieures.

Linux se veut interopérable, modulable et multiplateforme (avec de très nombreuses couches d'abstractions des couches inférieures du système). Tux est sa mascotte :

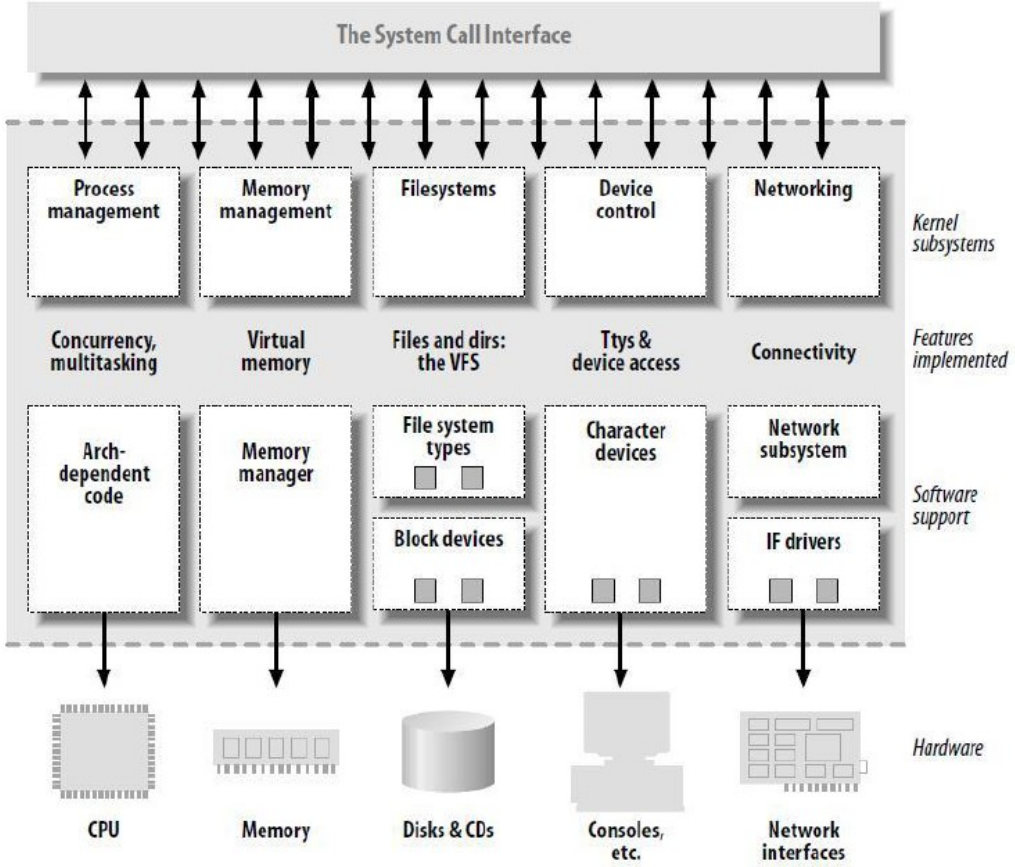
En fin 2020, on estime que depuis 2005 environ 21 000 développeurs issus de plus de 1 000 sociétés différentes ont contribué à l'écriture du kernel Linux.

Observons en 2020, les principales entreprises impliquées dans le développement de Linux : Intel, Red Hat, Huawei, Google, AMD, Linaro, Samsung …

Linux n’est qu’un noyau !

Observons les principaux services qu’il propose,

nous nous pencherons sur les détails plus tard.



Richard Matthew Stallman (rms) est considéré comme le fondateur du mouvement du logiciel libre, avec notamment la Free Software Fondation qu'il crée en 1985.

Il développe **GNU (GNU's Not UNIX)** en 1983, projet depuis repris par le GNU Project.

Avec **GNU**, Stallman cherche à créer une famille de composants logiciels tels qu'un compilateur C (**gcc**), un debugger (**gdb**), des bibliothèques système (**glibc** …), un éditeur de texte (**emacs**), un interpréteur de commande, une ré-implémentation des commandes standards …

GNU est donc un middleware, ou un OS sans kernel !

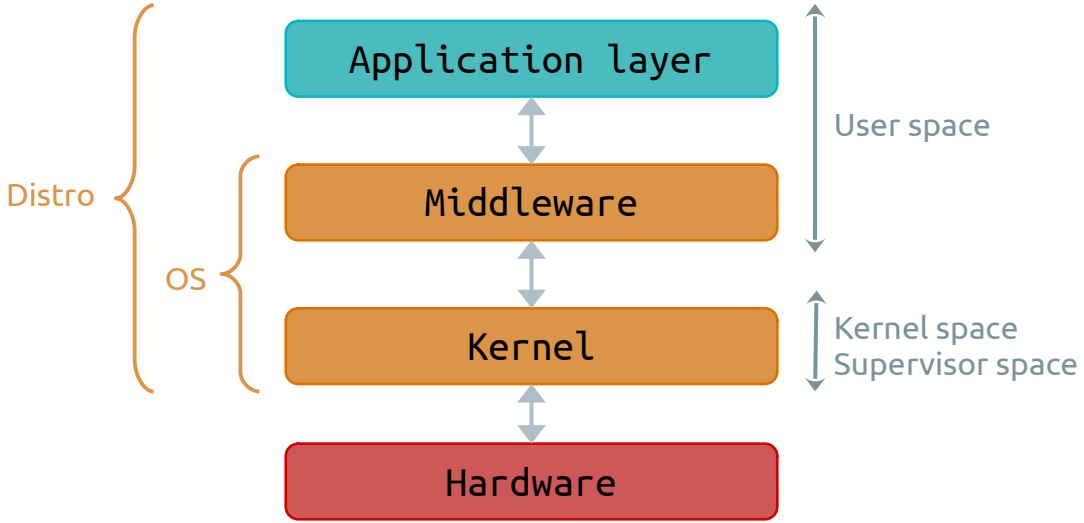
Rappelons qu'à l'époque UNIX n'était pas libre. Il fallut attendre 1990 avant d'avoir une premier kernel GNU nommé Hurd.

Une distribution GNU/Linux est une solution logicielle prête à l'emploi intégrant :

→ le kernel Linux

→ le middleware (GNU, BSD, commandes UNIX-like commandes, shell, GUI …)

→ des outils d'installation, d'administration et de mise à jour du système (`aptitude` sous Debian, `yum` sous Red Hat …).



Il existe un très grand nombre de distributions, sans compter le domaine de l'embarqué. Néanmoins, la grande majorité des distributions existantes descendent des trois suivantes :

Slackware : plus ancienne distribution encore en activité (1993)

Debian : éditée par une communauté de développeurs (1993)

Red Hat : éditée par l'entreprise Américaine du même nom (1994)

De nombreuses distributions sont dérivées des précédentes.
Voici les plus connues.

Ubuntu : distribution communautaire grand public éditée par Canonical et basé sur Debian. Souvent dépréciée des développeurs chevronnés pour son côté trop « *user friendly* ».

Linux Mint : orientée grand public, basée sur Debian et Ubuntu

Fedora : distribution communautaire supervisée et basée sur Red Hat.

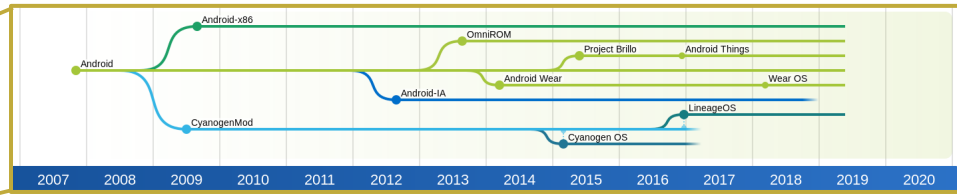
Les autres : SUSE (Novell), Gentoo, ArchLinux …
taper Linux From Scratch sur internet pour les curieux !

Distributions GNU/Linux

Debian

Ubuntu

Android



https://fr.wikipedia.org/wiki/Liste_des_distributions_GNU/Linux

Beaucoup de concepts rencontrés dans GNU/Linux sont issues de la philosophie UNIX, qui est largement répandue dans le domaine de l'ingénierie (*The Art of Unix Programming*, Eric S. Raymond).

La plupart de ces règles sont régies par le principe du rasoir d'Ockham ou principe de parcimonie :

Modularité :	Écrire des éléments simples reliés par de bonnes interfaces.
Clarté :	La clarté vaut mieux que l'ingéniosité.
Composition :	Concevoir des programmes qui peuvent être reliés à d'autres programmes.
Séparation :	Séparer les règles du fonctionnement et les interfaces du mécanisme.
Simplicité :	Concevoir pour la simplicité et n'ajouter de la complexité seulement par obligation.
Transparence :	Concevoir pour la visibilité de façon à faciliter la revue et le déverminage.
Robustesse :	La robustesse est l'enfant de la transparence et de la simplicité.

Parcimonie :	Écrire un gros programme seulement lorsqu'il est clairement démontrable que c'est l'unique solution.
Représentation:	Inclure le savoir dans les données, de manière à ce que l'algorithme puisse être bête et robuste.
La moindre surprise :	Pour la conception d'interface, réaliser la chose la moins surprenante.
Silence :	Quand un programme n'a rien d'étonnant à dire, il doit se taire.
Dépannage :	Si le programme échoue, il faut le faire bruyamment et le plus tôt possible.
Économie :	Le temps de programmation est cher, le préserver par rapport au temps de la machine.
Génération :	Éviter la programmation manuelle. Écrire des programmes qui écrivent des programmes autant que possible.
Optimisation :	Prototyper avant de fignoler. Mettre au point avant d'optimiser.
Diversité :	Se méfier des affirmations de « unique bonne solution ».
Extensibilité :	Concevoir pour le futur, car il arrivera plus vite que prévu.

ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

À partir de maintenant, vous avez compris ce qu'est Linux, et que dans une discussion non technique « Linux » veut tout et rien dire. Alors précisons.

Techniquement, Linux est un kernel libre.

Mais dans le langage courant, cela désigne n'importe quelle distribution basée sur l'association de GNU et Linux (GNU/Linux).

Clarifions le vocabulaire utilisé pendant ce module :

Kernel = noyau = Linux

OS = kernel + Middleware = Linux + GNU

Distribution = OS + outils d'administration (Ubuntu, Red Hat, Fedora, Raspbian, …)

LINUX POUR L'EMBARQUÉ

Informatique vs Embarqué

Pourquoi Linux

ENSI CAEN

ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

Le monde de Linux pour l'embarqué est par essence différent du monde du PC.

En effet, le principe même du domaine de l'embarqué est de fournir une solution logicielle spécifique, sur une plateforme matérielle précise (grande hétérogénéité des plateformes et développement en milieu contraint, ressources CPU et mémoire) pour une application particulière …

Ceci est à comparer à l'universalité des solutions logicielles (multiplateformes) dans le monde des ordinateurs (processeurs généralistes) et la philosophie même des systèmes UNIX.

Computer-world Linux

Embedded Linux
faster, lighter, optimized, ...

Le domaine des systèmes embarqué offre une grande richesse d'architectures matérielles différentes (malgré un marché dominé par les architectures ARM) et peut exiger selon les volumes en jeux de développer des applications en milieux contraints. Observons les principales optimisations associées au domaine.

Vitesse d'exécution (CPU)

- temps de boot (bootloader, services réseaux et interfaces I/O minimalistes, décompression noyau, minimiser taille et services kernel et user space …) ;
- options de compilation (architecture dépendant) ;
- exécutables GNU/UNIX et librairies optimisées (Busybox, uClibc …) ;
- conception logicielle (mono-processus, inlining …) ;
- outils de trace et de profilage …

Empreinte mémoire

- contradictions avec le domaine de l'optimisation CPU (*uninlining*, options de compilation `-Os`, compression kernel et *file system*, librairies partagées …) ;
- taille optimale du kernel et de la distribution (temps de chargement en RAM rapide) ;
- utiliser *initramfs* au lieu de *initrd* ;
- exécutables GNU/UNIX et librairies optimisées (Busybox, uClibc … en désaccord avec la philosophie UNIX) ;
- *stripping* ;
- gestionnaire dynamique de paquets souvent inutile …

Consommation

- option d'alimentation ;
- fréquence de travail du cœur ;
- périodicité de préemption du kernel …

Coût matériel

Optimiser les aspects précédemment cités permet de jouer grandement sur le design de l'architecture matérielle finale (coût processeur, RAM DDR, *Mass storage* MMC/SDcard, eMMC, …). Aspects non négligeables en fonction des volumes en jeu et marchés ciblés.

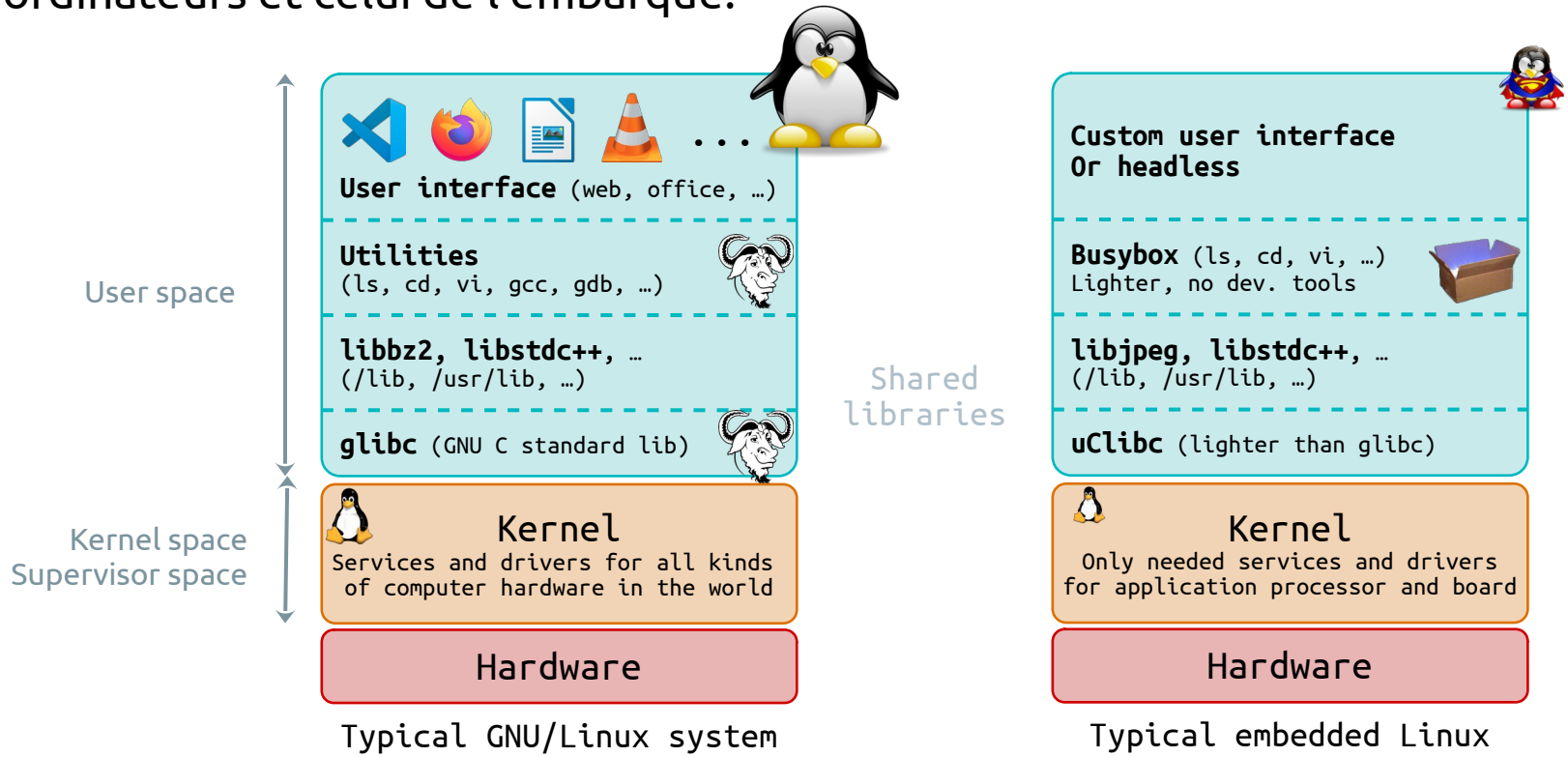
Vitesse d'exécution (CPU)

Permet de travailler avec un processeur à plus faible coût et peut jouer sur la consommation de l'application finale (coût de la batterie et du système d'alimentation de la plateforme).

Empreinte mémoire

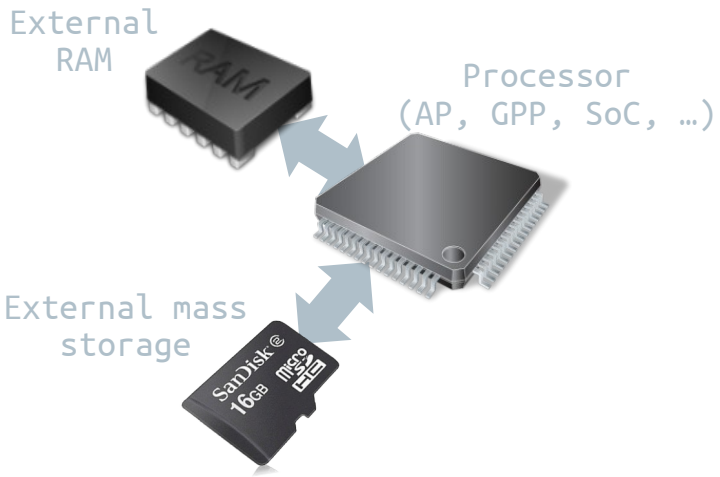
Moins de *swapping* et un usage réduit de la RAM (mémoire vive et de stockage de masse moins coûteuse), besoin de moins de cache processeur (processeur moins coûteux).

Observons l'architecture typique d'un système GNU/Linux dans le monde des ordinateurs et celui de l'embarqué.

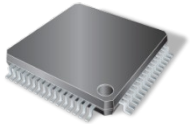


Prenons quelques exemples du coût global d'un processeur associé à sa mémoire (principale et secondaire), sans interfaces de communication, pour une application dans l'embarqué. Les coûts sont donnés à titre indicatif (contre-exemples nombreux).

Typical embedded Linux hardware
~ $4-5 < cost < $30-40



Typical RTOS hardware
~ $2-3 < cost < $10



Processor (MCU, DSP, SoC, …)

Typical OS-less hardware
~ $0.10 < cost < $2-3

Processor (MCU, DSP, …)

ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

Contraintes des systèmes propriétaires

- Les coûts de license sont élevés, voire très élevés
- L'éditeur du logiciel peut disparaître, et le support technique avec
- Ou alors, paiement d'un surplus pour s'assurer l'accès aux sources
- Logiciel pas sur-mesure (fonctionnalités superflues ou peu adaptées)

ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

Avantages de l'open source

- Redistribution sans royalties
- Disponibilité du code source
- Modification du code source pour adapter une solution sur-mesure
- Pour Linux, la taille du projet assurer une pérennité des solutions sur le très long terme

Et Linux en particulier

- Fiabilité (robustesse, qualité)
- Performances
- Portabilité matérielle (architectures processeurs)
- Adaptabilité (systèmes de fichiers, modularité, ...)
- Support technique

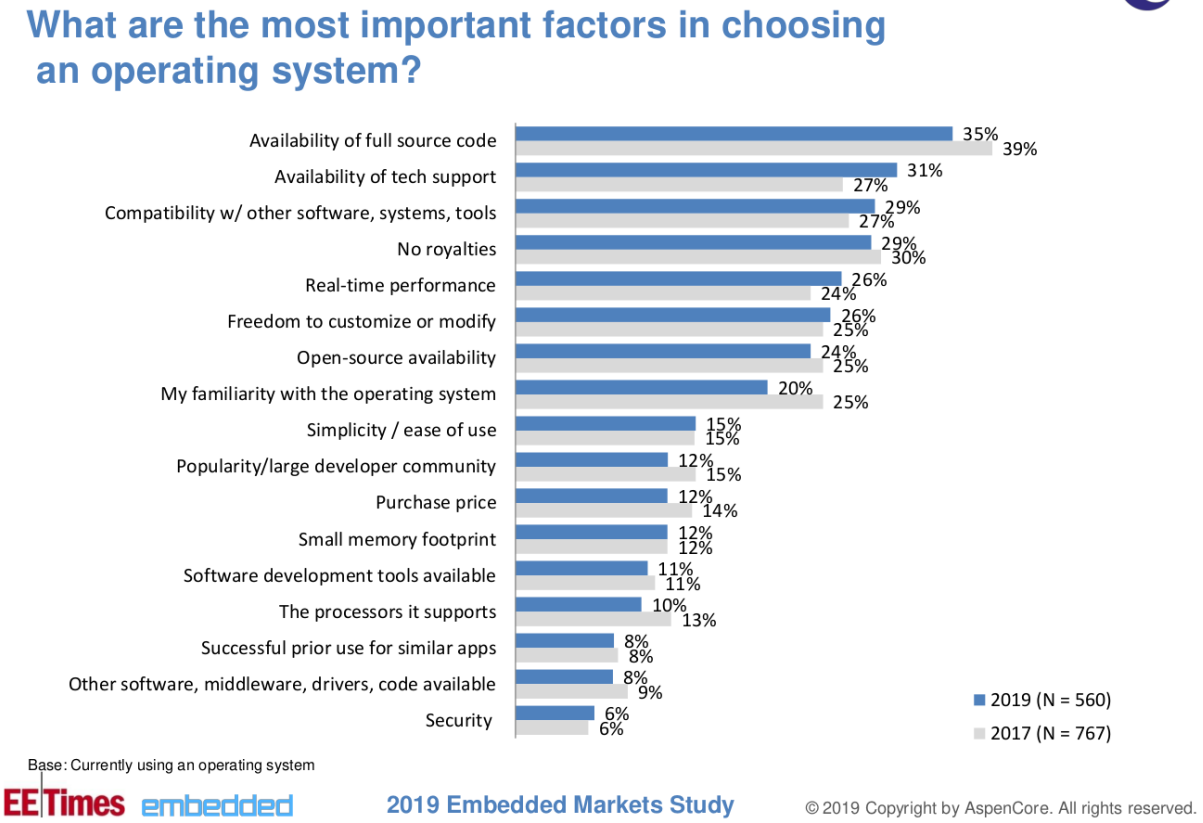
Critères de sélection d'un OS, vu par les pros.

Source :

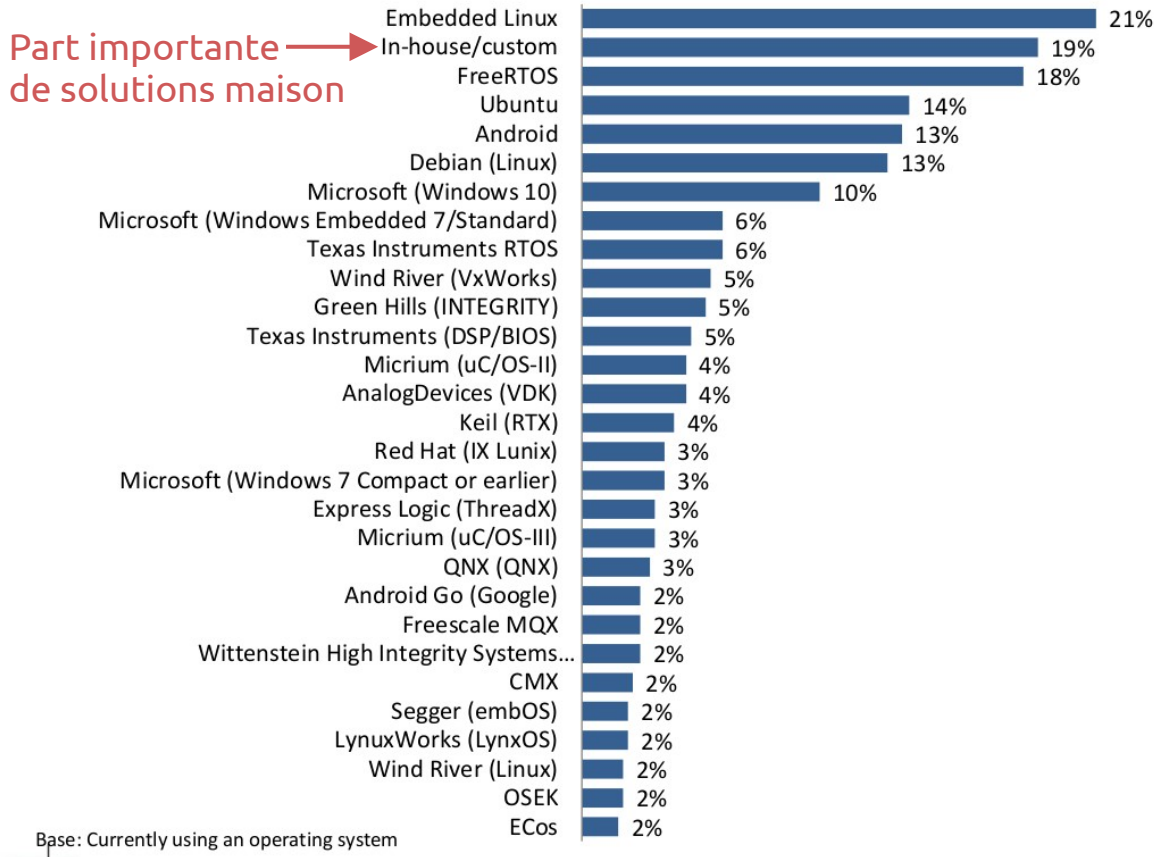
2019 Embedded Markets Study

Integrating IoT and Advanced Technology Designs, Application Development & Processing Environments March 2019

Presented by : EETimes, embedded



Part de marché des OS pour l'embarqué



Source :

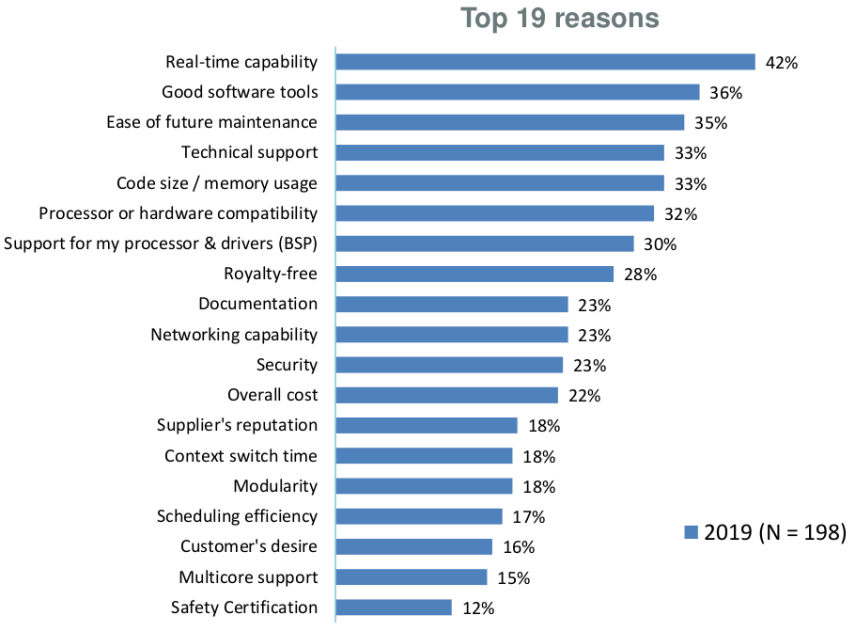
2019 Embedded Markets Study

Integrating IoT and Advanced Technology Designs, Application Development & Processing Environments March 2019

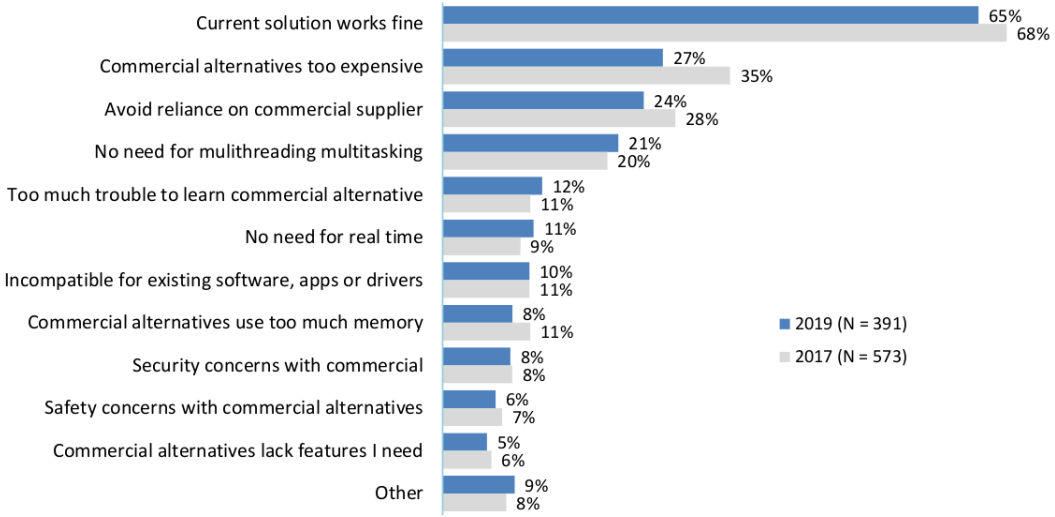
Presented by : EETimes, embedded

Pour et contre un OS propriétaire

Which factors most influenced your decision to use a commercial operating system?



What are your reasons for not using a commercial operating system?



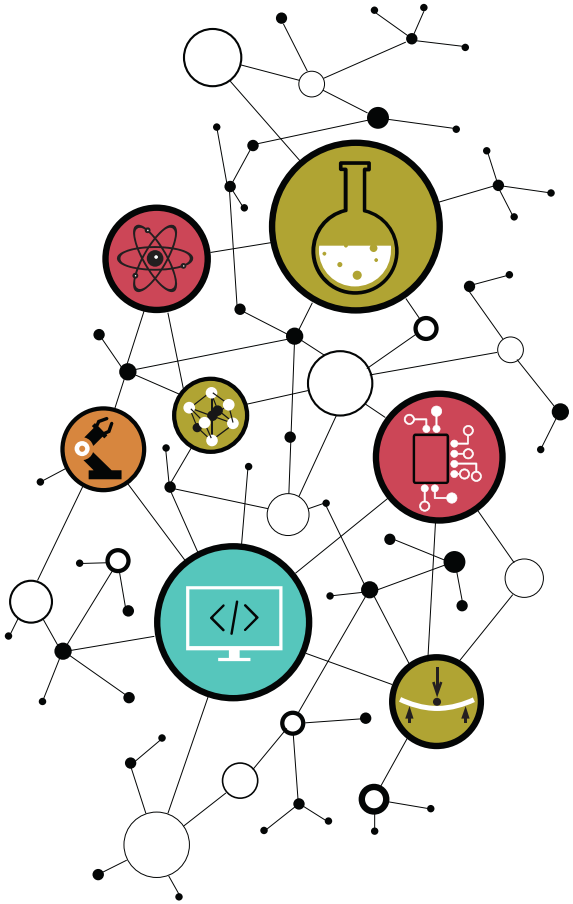
Quand passer sous Linux pour l'embarqué ?

- Quand il faut gérer un système "complexe" (trop complexe pour un RTOS) : Par exemple, pilotage de plusieurs librairies réseau (WIFI, BT, BLE, IP, etc) voire avec une interface graphique évoluée.
- Quand il faut gérer une IHM avancée

Quand éviter Linux ?

- Sur des applications industrielles à fort volume
- Sur des applications avec des contraintes temps réel très lourdes
- Sur des applications faiblement évolutive à architecture système "simple"
- Matériel inadapté (mémoires trop faibles, pas de MMU (sauf uClinux), …)
- Si la license GPL/LGPL ne correspond pas au projet
- Si on doit garantir la conformité à des standards (aéronautique, ...)

RESSOURCES EN LIGNE



Sites références concernant le projet Linux (notamment en France) :

http://www.linux.org/ , http://linuxfr.org/ , http://www.linux-france.org/

Archives du kernel Linux :

https://www.kernel.org/

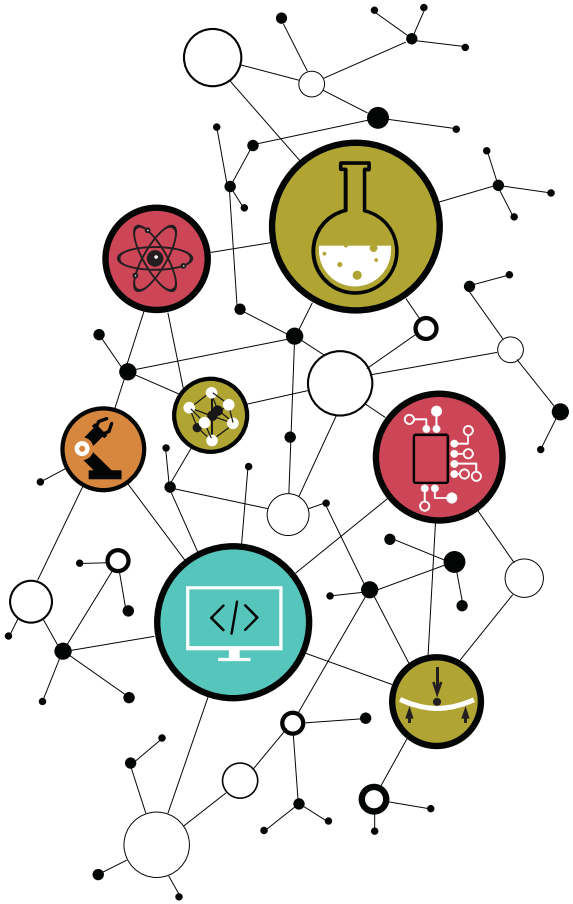
Site TI pour Linux sur architecture OMAP :

http://linux.omap.com/

bootlin, société française de développement, services et formation autour de Linux embarqué :

https://bootlin.com/

RESSOURCES EN LIGNE

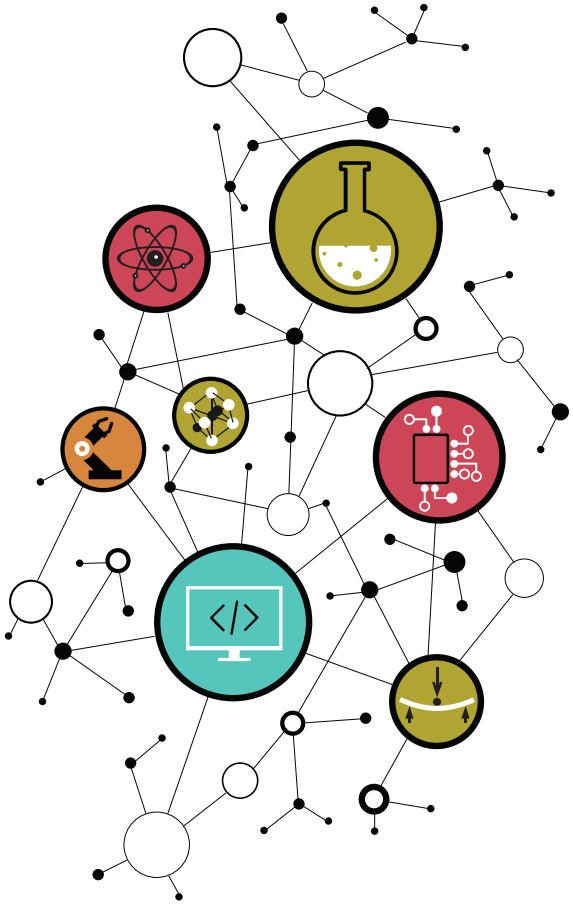


Livre Linux Embarqué de Pierre Ficheux et édité par eyrolles :
http://www.eyrolles.com/Informatique/Livre/linux-embarque-9782212134827

Cours en ligne proposé par l'ENST Bretagne de Jean-Marie Gilliot :
http://public.enst-bretagne.fr/~jgilliot/linux/LinuxEmbarque.html

Cours en ligne proposé par l'ENSEIRB-MATMECA de Patrice Kadionik :
http://kadionik.vvv.enseirb-matmeca.fr/

CONTACT



Dimitri Boudier – PRAG ENSICAEN

dimitri.boudier@ensicaen.fr

Avec l'aide précieuse de :

- Hugo Descoubes (PRAG ENSICAEN)