UNIVERSITÉ
CÔTE D'AZUR

Introduction aux Systèmes et Logiciels Embarqués

Présentation: **Stéphane Lavirotte**
Auteurs: ... et al*

(*) Cours réalisé grâce aux documents de :
Christophe Blaess, Stéphane Lavirotte, Jean-Paul Rigault

Mail: Stephane.Lavirotte@univ-cotedazur.fr
Web: http://stephane.lavirotte.com/
Université Côte d'Azur

UNIVERSITÉ
CÔTE D'AZUR

Présentation du Cours

Préface

UNIVERSITÉ CÔTE D'AZUR

Objectifs du Cours

- ✓ **But**
  - Faire le choix d'une architecture en fonction des contraintes d'un projet
  - Comprendre les mécanismes de démarrage d'un système
  - Mettre en œuvre un système embarqué « *from scratch* »
  - Déployer les applications adaptées sur la cible

- ✓ **Axé sur l'expérimentation**
  - Pendant 12 semaines: 1h de cours, 3h de TD + Evals
  - Travail sur machine virtuelle et sur matériel
    - Raspberry Pi, Arduino
  - Illustrations et études de cas sur Linux

UNIVERSITÉ CÔTE D'AZUR

Systèmes Embarqués

- ✓ Aujourd'hui, la plupart des systèmes électroniques utilisés sont des systèmes embarqués:
  - Quotidien: téléphones mobiles, télévision, électroménager...
  - Santé: moniteur cardiaque, pompe à glucose, pacemakers, ...
  - Transport: automobile, avion, bateau, bus...
  - Objets connectés: montre, pèse-personne, brosse à dents...
- ✓ Contraintes sur:
  - La taille (intégration, entrées-sorties)
  - La consommation électrique (autonomie, puissance de calcul)
  - Le coût (quantité de mémoire, type de processeur)
- ✓ Mais de différents types...

UNIVERSITÉ CÔTE D'AZUR

Différents Systèmes Embarqués

- ✓ Systèmes vs Systèmes critiques
  - « Business critical » → time-to-market
    - Téléphone, audio, TV, DVD, jeux, …
  - « Mission critical » → qualité supérieur
    - Trajectoire et altitude, imagerie, transmission, …
  - « Life critical » → validation et certification
    - Pacemakers, contrôle de glucose, robots chirurgiens, …
  - « Safety critical » → validation et certification
    - Pilotage, frein, distribution électronique, carburant, …

- ✓ Différentes contraintes
  - Tout peut-être vu comme critique
  - Mais nous ne traiterons pas de la validation de programme

UNIVERSITÉ CÔTE D'AZUR

Bibliographie

✓ Ouvrages

- **Pierre Ficheux, Eric Bénard,** *Linux embarqué. Nouvelle étude de cas - Traite d'OpenEmbedded*, Eyrolles, 2012
- **Donald Norris**, Projets créatifs avec Raspberry Pi, fév 2014
- **Chistophe Blaess**, Solution temps réel sous Linux, 3ème édition, Eyrolles, nov 2019

✓ Cours

- Experts Linux embarqué: https://bootlin.com/

✓ Journaux grand public

- Open Silicium: http://www.opensilicium.com/
- GNU Linux Magazine France: http://www.gnulinuxmag.com/
- GNU Linux Pratique France: http://www.linux-pratique.com/

UNIVERSITÉ
CÔTE D'AZUR

Microprocesseur
Microcontrôleur

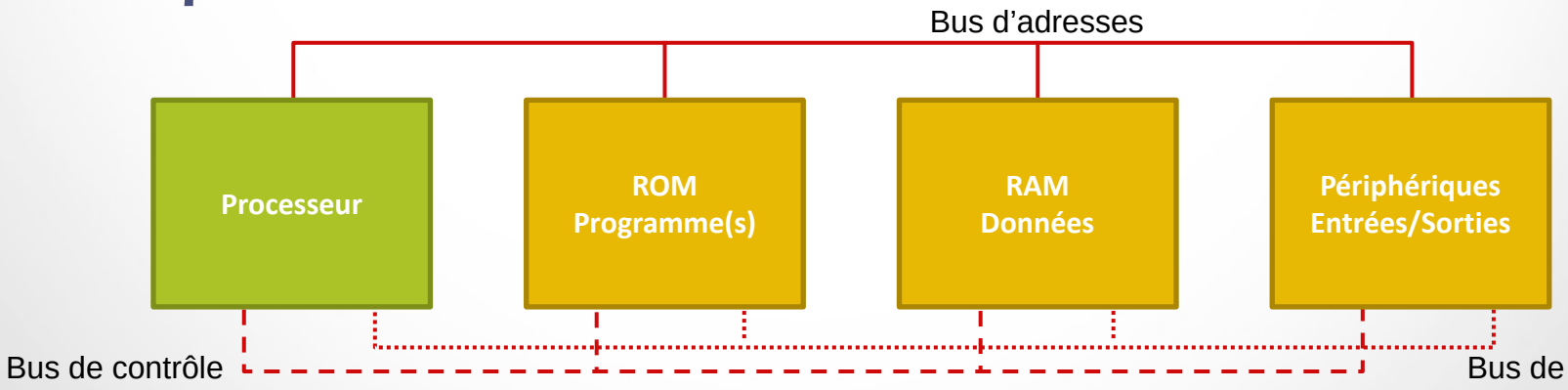
Qui est qui ?

Niveau matériel

UNIVERSITÉ CÔTE D'AZUR

Processeur et Traitement de l'Information

- ✓ Processeur: élément central d'un système informatique
  - Interprète les instructions et Traite les données
- ✓ Besoins d'éléments complémentaires
  - Horloge pour le cadencer
  - Mémoire pour l'exécution des programmes (ROM) pour le stockage (RAM)
  - Périphériques
- ✓ Bus pour relier ces entités



UNIVERSITÉ CÔTE D'AZUR

Microprocesseur Microcontrôleur

- ✓ Le microprocesseur
  - Intégration dans des circuits distincts
  - Nécessité de prévoir l'interconnexion (bus, câblage)
  - La place occupée par ces composants séparés est plus importante
  - Plus de consommation et de chaleur dégagée
  - Coût financier
- ✓ Le microcontrôleur
  - Rassemble ces éléments sur un seul circuit intégré
  - Composant autonome, capable d'exécuté des programmes sur sa ROM
  - Améliore l'intégration et le coût
  - Moins de capacités que le microprocesseur

UNIVERSITÉ CÔTE D'AZUR

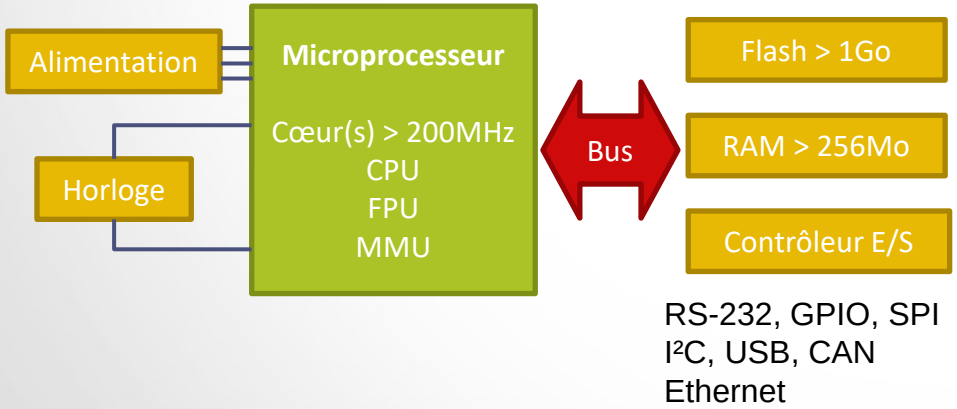
Comparaison µP - µC (1/2)

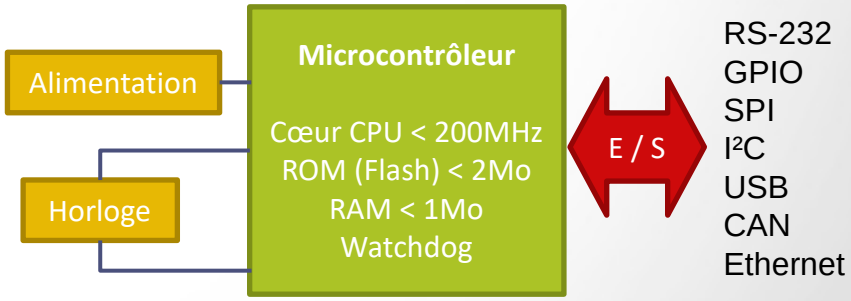
Microprocesseur

- ✓ Electronique complexe
- ✓ Entrées-sorties par des contrôleurs externes
- ✓ Utilisation d'un OS

Microcontrôleur

- ✓ Electronique simple
- ✓ Déterminisme
- ✓ Fiabilité fonctionnement
- ✓ Généralement sans OS





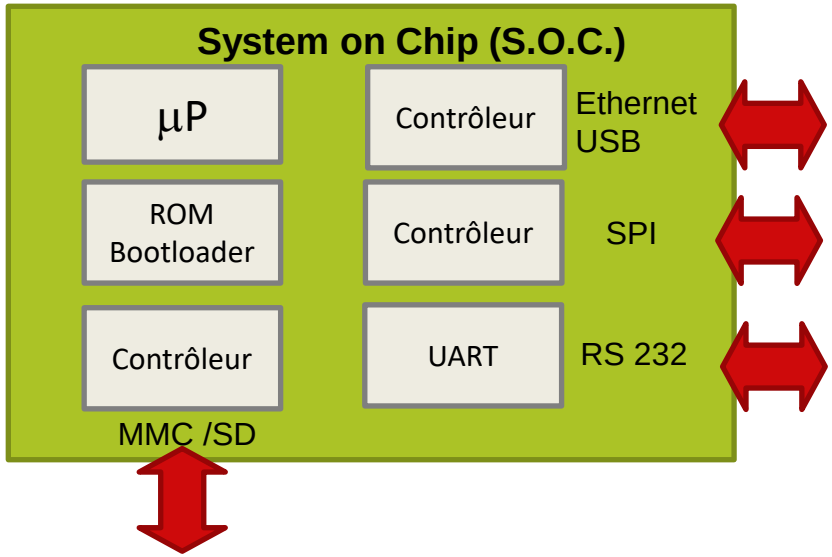
UNIVERSITÉ CÔTE D'AZUR

Comparaison µP - µC (2/2)

	Microprocesseur	Microcontrôleur
Coût moyen	> 20 €	< 10€
PCB support	Complexe (6 couches)	Simple (2 couches)
Alimentation	Multiples – 3.3V / 5V / 12V	Simple – 3.3V
Volume de code métier	Plusieurs dizaines de Mo	Quelques Ko
Environnement développement	Libre et gratuit si développement Linux	Généralement propriétaire (coût élevé)
Mise au point de code	Classique grâce à l'OS	Complexe, débuggueur spécifique
Déploiement, MàJ	Simple	Complexe
Protection code métier	Difficile	Facile (fusibles)

UNIVERSITÉ CÔTE D'AZUR

System on Chip (S.o.C.)

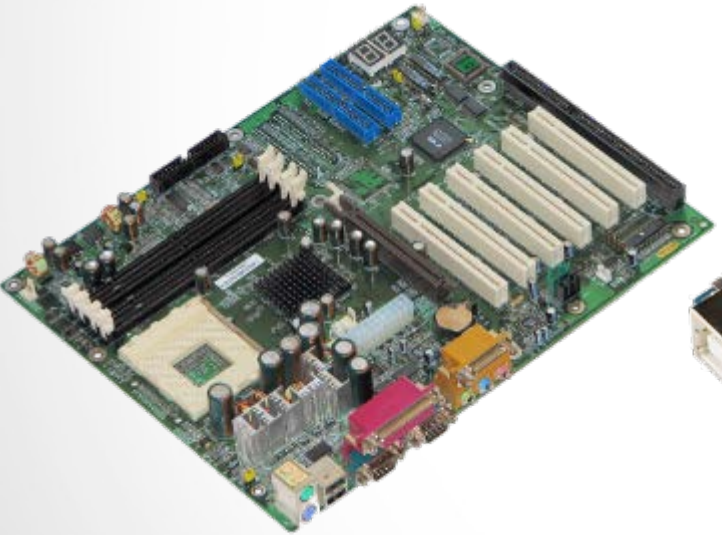


- ✓ Contrôleurs d'entrées-sorties déjà incorporés
- ✓ Intégration électronique plus complexe
- ✓ Souvent peu d'entrées sorties industrielles (CAN) ou analogiques (ADC/DAC, PWM)

UNIVERSITÉ
CÔTE D'AZUR

Qui est qui ?

✓ µP – µC – SoC ?



« Ordinateur de bureau »
Microprocesseur

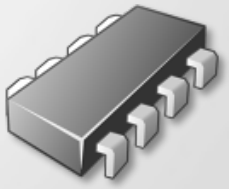
Arduino Uno
Microcontrôleur

Raspberry Pi
SoC

UNIVERSITÉ CÔTE D'AZUR

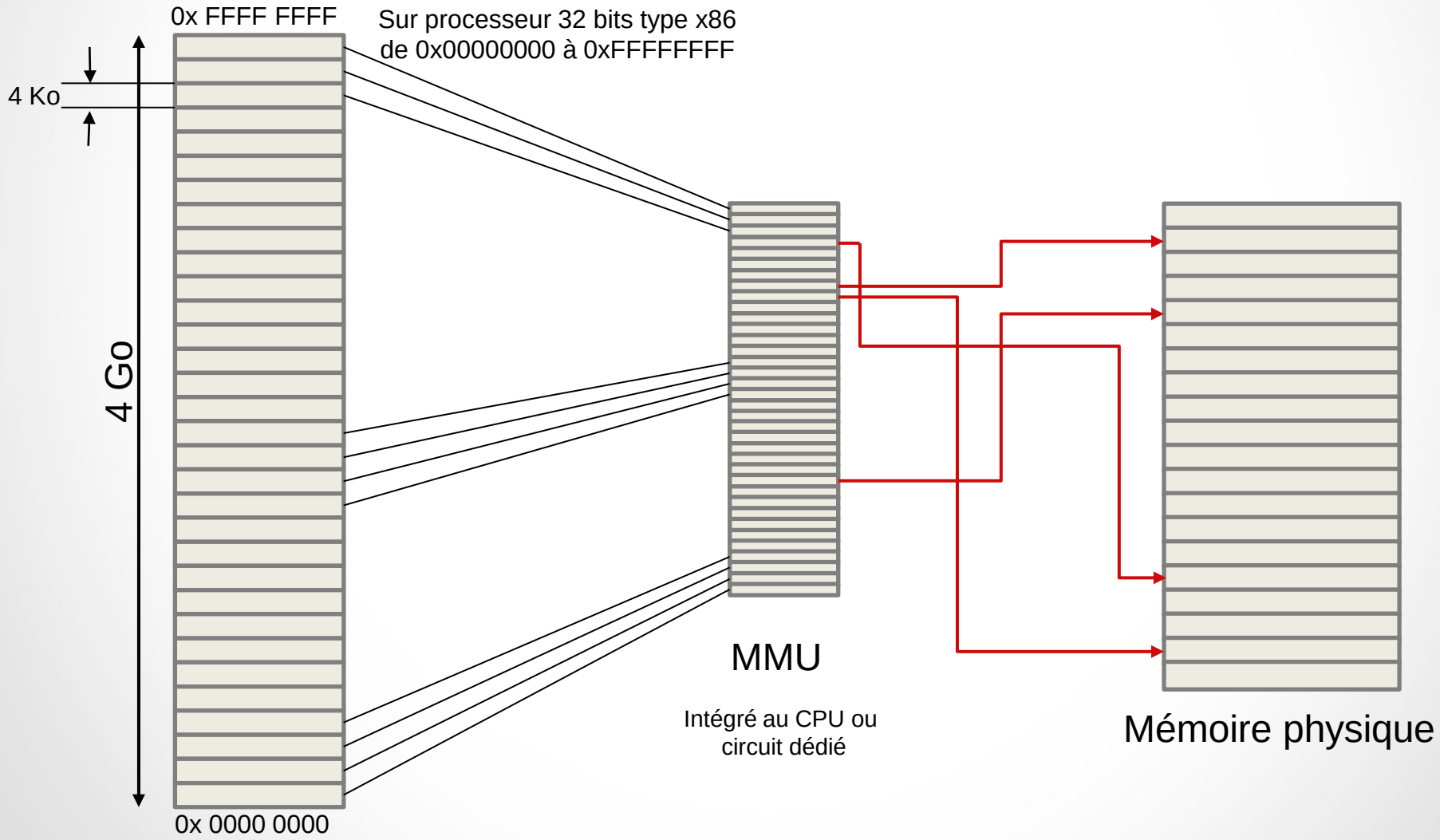
Gestion Mémoire Virtuelle

- ✓ Les adresses mémoires gérées dans les processus ne sont pas directement des références à des adresses physiques
  - Chaque processus a son espace d'adressage virtuel
- ✓ Les adresses virtuelles sont traduites en adresses physiques
  - Le circuit MMU opère la conversion à chaque référence
    - A l'aide d'un ensemble de registres qui désignent les tables de conversion du processus courant
    - Une table de conversion est associée à chaque processus
  - Lors du changement de contexte (= processus courant)
    - chargement des registres du MMU avec de nouvelles adresses de tables



UNIVERSITÉ CÔTE D'AZUR

Mémoire Virtuelle et MMU

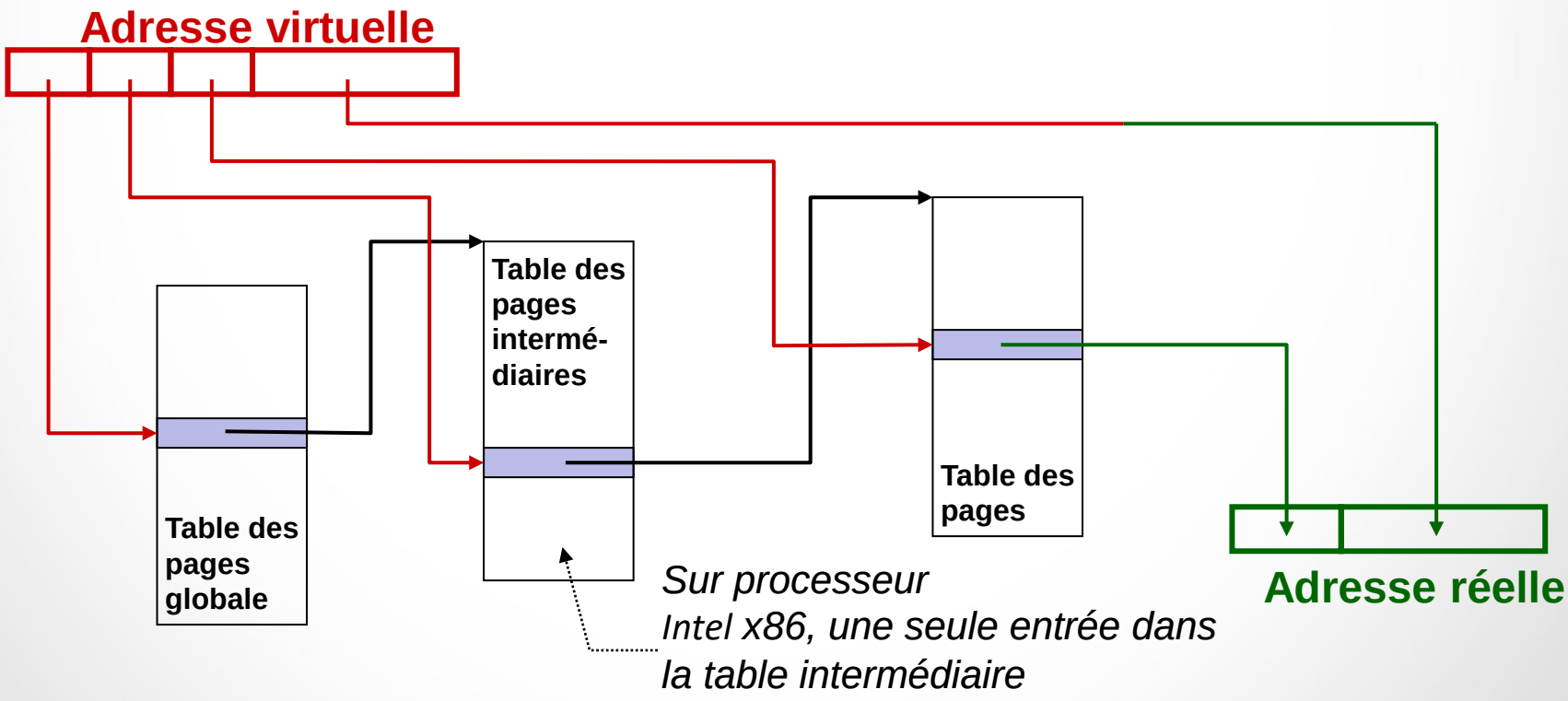


Espace d'adressage virtuel

UNIVERSITÉ CÔTE D'AZUR

Zoom Gestion Mémoire Virtuelle

- ✓ Modèle de pagination indépendant du processeur
- ✓ Table des pages à multiples niveaux



UNIVERSITÉ CÔTE D'AZUR

Critères de Comparaison des Systèmes Embarqués

- ✓ Classe de processeur:
  - Microprocesseur / Microcontrôleur
- ✓ Gestion de la mémoire:
  - MMU / sans MMU
- ✓ Système d'exploitation:
  - OS / sans OS
- ✓ Type de système d'exploitation
  - Normal / Temps Réel

- ✓ Tous ces paramètres, en plus des plus classiques (fréquences, quantité de mémoire, ...), permettent de caractériser un système embarqué

Système d'Exploitation

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

UNIVERSITÉ CÔTE D'AZUR

Caractéristiques d'un Système d'Exploitation

- ✓ Un système d'exploitation est un ensemble de programmes qui dirige l'utilisation des capacités de la machine
- ✓ But:
  - Faire une abstraction du matériel
  - Gérer le temps (temps partagé, temps réel)
  - Gérer la distribution (entre les processeurs, les mémoires, les périphériques) pour augmenter l'efficacité et abstraire
- ✓ Fonctionnalités
  - Servir les requêtes des processus
    - **Appels systèmes**: Read, Write, Open, ...
  - Traiter les exceptions matérielles dues aux processus
    - **Déroutements**: Division par 0, Débordement de pile, ...
  - Gérer les interruptions matérielles
    - **Interruptions**: clavier / souris, réseau, ...
  - Fournir un ensemble de services spécifiques
    - **Assurer des tâches d'entretien du système** (swap, caches, pages, ...)

UNIVERSITÉ CÔTE D'AZUR

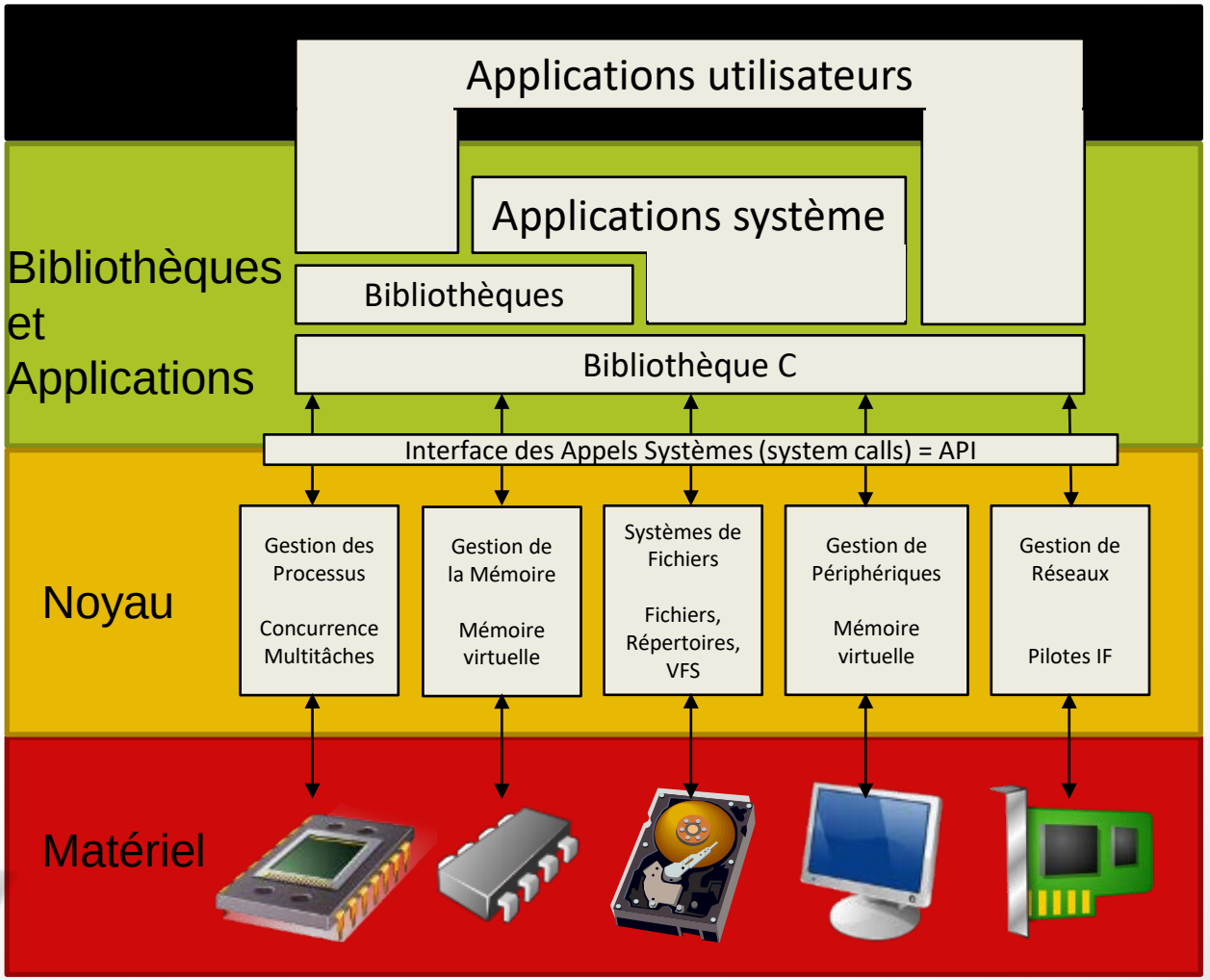
Eléments d'un Système d'Exploitation

- ✓ 3 éléments principaux
  - Le Noyau
    - Programme qui est le premier à s'exécuter après le chargeur
    - Fournit les abstractions pour la gestion des processus de la mémoire, des systèmes de fichiers, ...
    - Voir cours SI5: « *Systèmes d'Exploitation avancés* »
  - La/Les librairies
    - Bibliothèque(s) standardisées de fonctionnalités pour les programmes utilisateurs
    - Librairie C (libc), librairie math (libm)
    - Voir cours SI3: « *Programmation Systèmes – Posix* »
  - Les programmes utilitaires
    - Ensemble de programmes outils permettant de manipuler le système à base de commandes « basiques »
    - Ex: `shell`, `cp`, `rm`, `mount`, ...
    - Voir cours PeiP 1: « *Environnement Informatique 1* »

UNIVERSITÉ CÔTE D'AZUR

Architecture générale d'un Système d'Exploitation

XNU



UNIVERSITÉ CÔTE D'AZUR

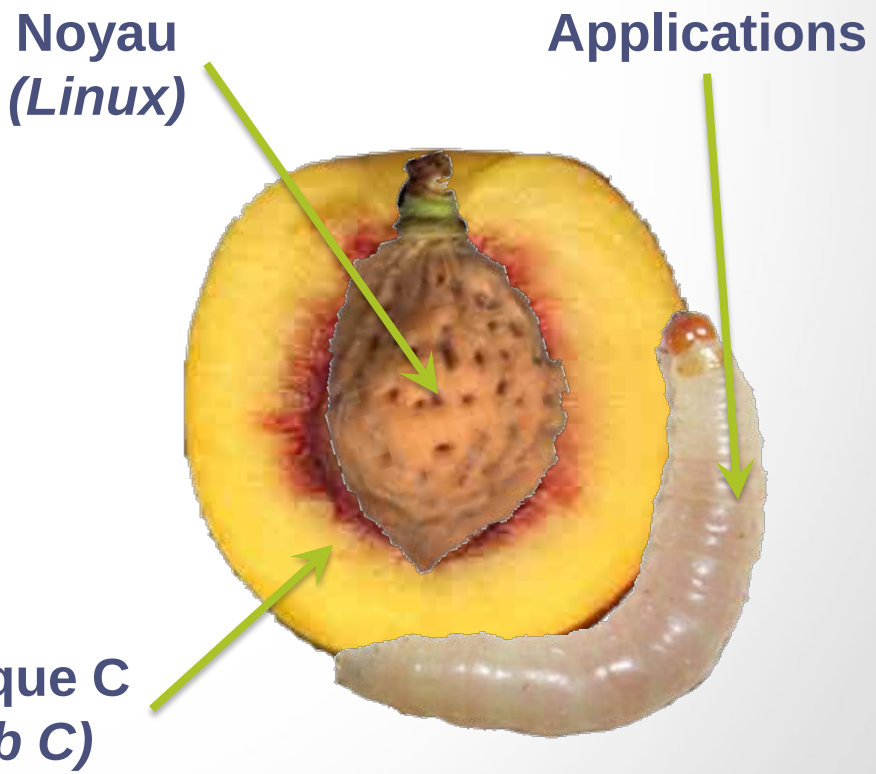
Noyau vs Système d'Exploitation

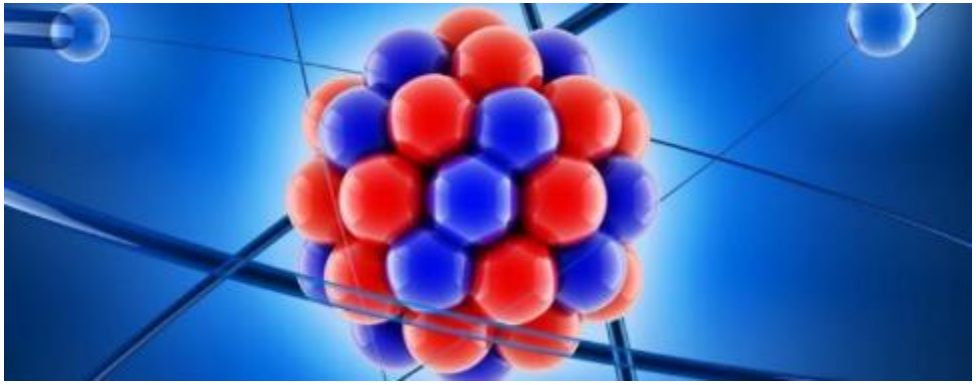
- ✓ Ne pas confondre:
  - Noyau

  et
  - Système d'exploitation

- ✓ Exemple avec « Linux »
  - Le Noyau
    - Linux
  - Le Système d'Exploitation
    - GNU

- ✓ Donc on parle de:
  - GNU / Linux





Le Noyau

D'abord et avant tout un noyau

UNIVERSITÉ CÔTE D'AZUR

Un Noyau: c'est quoi ?

- ✓ Avant tout c'est un programme:
  - Réside sur le disque
    - Par exemple pour Linux dans `/vmlinuz` ou `/boot/vmlinuz`
  - Est le « premier » élément chargé (après le bootloader)
- ✓ Mais un programme spécial
  - S'exécute au plus proche du matériel (juste au dessus du BIOS)
  - Accès privilégié au matériel
  - Implémente l'abstraction des processus
    - Mais ce n'en est pas un lui-même
- ✓ Son rôle:
  - Fournir les abstractions et les interfaces aux accès matériels
  - Gestion a minima
    - des processus, de l'ordonnancement, des IPC
    - de la mémoire virtuelle

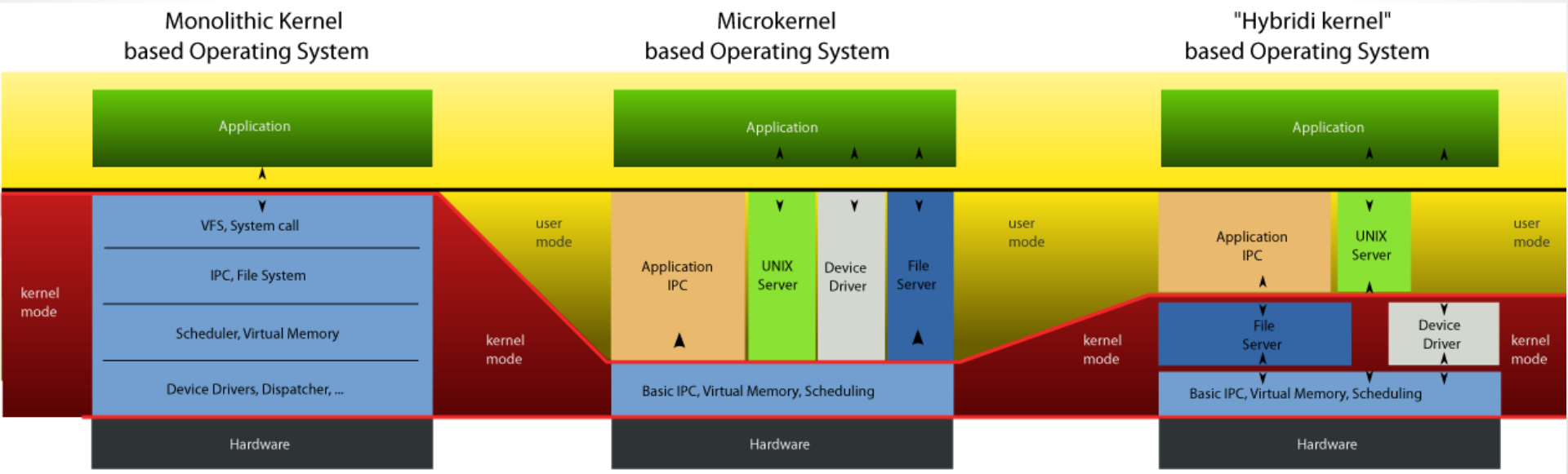
UNIVERSITÉ CÔTE D'AZUR

Différents types de Noyau (1/2)

- ✓ Noyau Monolithique
  - Fournit tous les services (prog. unique, modulaire ou non)
  - Tout s'exécute en *mode noyau*
  - Ex. non modulaire: DOS, Windows 9x, MacOS <9, ...
  - Ex. modulaires: Linux > 1.2, BSD, Solaris, ...
- ✓ Micro Noyau
  - Fournit les services minimaux
    - gestion des processus, de la mémoire et des IPCs
  - Les autres services sont fournis par des programmes utilisateurs
  - Ex.: Mach (Mac OS X), L4 (GNU/Hurd)
- ✓ Hybride
  - Combinaison du meilleur des deux mondes ?
  - Ex.: Windows NT (Windows 7, 10), XNU (Mac OS X, iOS)
- ✓ http://fr.wikipedia.org/wiki/Noyau_de_système_d'exploitation

UNIVERSITÉ CÔTE D'AZUR

Différents types de Noyau (2/2)



UNIVERSITÉ CÔTE D'AZUR

Comparaison des Types de Noyau

- ✓ Noyau Monolithique
  - Plus facile à écrire
  - Moins élégant que les micro noyau
  - Plus performants
- ✓ Micro Noyau
  - Très intéressant en théorie, plus difficile en pratique
  - Plus résistant aux bugs (donc plus sûre)
- ✓ Hybride
  - Combinaison du meilleur des deux mondes ?

- ✓ Combats virulents entre Monolithique et Micro
  - Tanenbaum vs Torwald
  - Mais l'histoire semble faire ressortir l'hybride et le monolitique

UNIVERSITÉ
CÔTE D'AZUR

Un exemple de Noyau: Linux

Illustration avec le

2.6 ≤ Noyau Linux ≤ 5.10

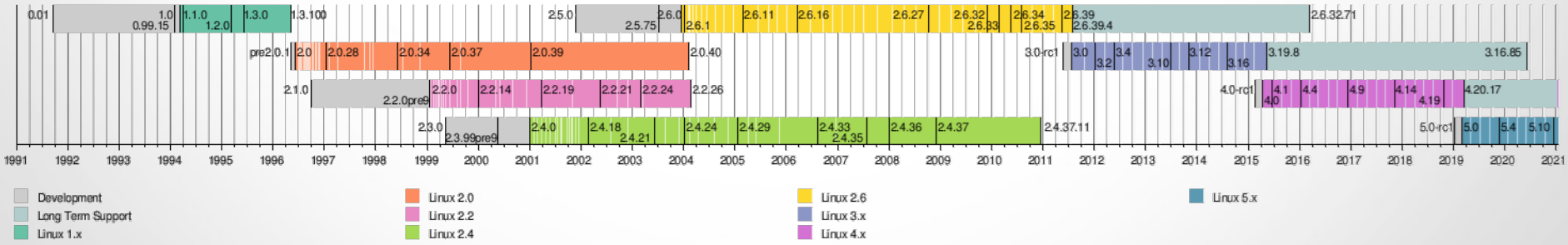
UNIVERSITÉ CÔTE D'AZUR

Historique

- ✓ 1991: Création de Linux à partir de zéro par Linus Torwald en 6 mois (étudiant à l'Université d'Helsinki)
- ✓ 1991: Distribution de Linux sur Internet par l'auteur. Des programmeurs du monde entier contribuent
- ✓ 1992: Linux est distribué sous licence GNU GPL
- ✓ 1994: Sortie de Linux 1.0
- ✓ 1994: Création de la société RedHat
- ✓ 1995: GNU/Linux se répandent sur les serveurs
- ✓ 2001: IBM investit 1 milliard de dollars dans Linux
- ✓ 2002: Adoption de Linux dans de nombreux secteurs
- ✓ ... et depuis l'aventure ne fait que continuer

Historique des Versions du Noyau

- ✓ Historique des versions du Noyau
  - 1991 - 1996: 0 et 1
  - 1996 - 2004: 2.0
  - 1999 - 2004: 2.2
  - 2001 - 2011: 2.4
  - 2004 - 2016 : 2.6
  - 2011 - ... : 3.x
  - 2015 - ... : 4.x
  - 2019 - ... : 5.x



UNIVERSITÉ CÔTE D'AZUR

Numérotation des versions

- ✓ Jusqu'en 2003, une nouvelle version stable tous les 2 à 3 ans (2.0, 2.2, 2.4, 2.6)
- ✓ Depuis 2003, une version stable toutes les 10 semaines
  - Version 2.6 (Déc 2003) à 2.6.39 (May 2011)
  - Version 3.0 (Juil 2011) à 3.19 (Fév 2015)
  - Version 4.0 (Avr 2015) à 4.20 (Déc 2018)
  - Version 5.0 depuis Mars 2019
- ✓ Les fonctionnalités sont ajoutées au noyau au fur et à mesure. Depuis 2003, les développeurs noyaux les introduisent sans modification massive de la branche de développement
- ✓ Pour chaque version, mise à jour et sécurité: 5.0.1, 5.0.2, etc.

UNIVERSITÉ CÔTE D'AZUR

Quel Noyau Choisir ?

- ✓ De très nombreuses versions du noyau disponibles
- ✓ Quelle version choisir ?
  - La plus récente n'est pas forcément la meilleure ou la plus adaptée
  - Il est nécessaire de croiser plusieurs informations
    - De quelles nouvelles fonctionnalités ai-je besoin ? http://fr.wikipedia.org/wiki/Noyau_Linux
    - Consulter sur le site http://www.kernel.org les versions qui sont mises en avant (long-term support)
    - Vérifier avec des distributions qui ont des noyaux sur de longues périodes (comme Debian par exemple)
- ✓ Ces questions doivent vous conduire à orienter votre choix

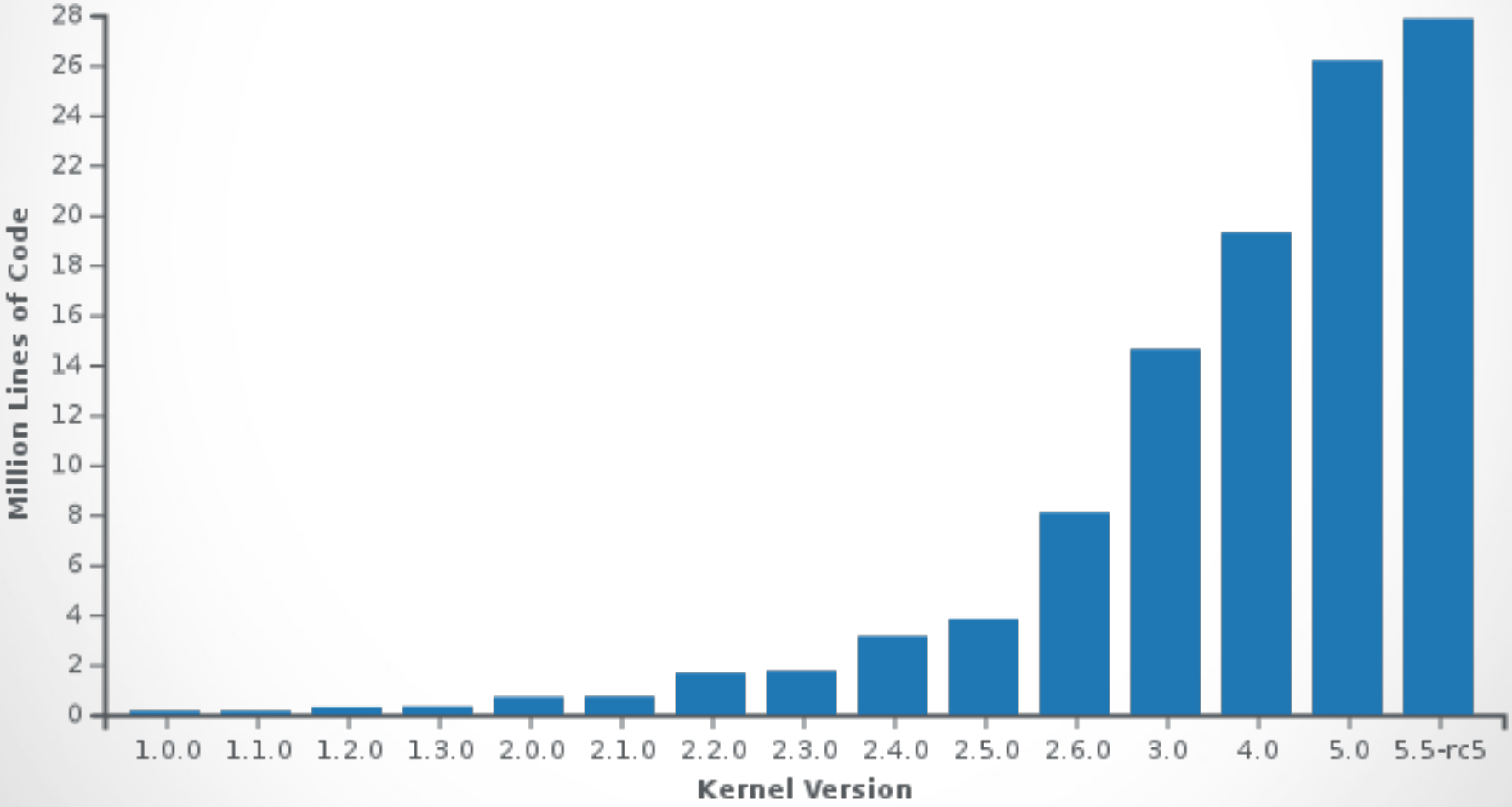
UNIVERSITÉ CÔTE D'AZUR

Activités de Développement sur le Noyau Linux et Contributeurs

- ✓ *Une petite vidéo vaut mieux qu'un long discours*

UNIVERSITÉ
CÔTE D'AZUR

Linux SLOC



1994 1996 1999 2001 2004 2011 2015 2019 2020

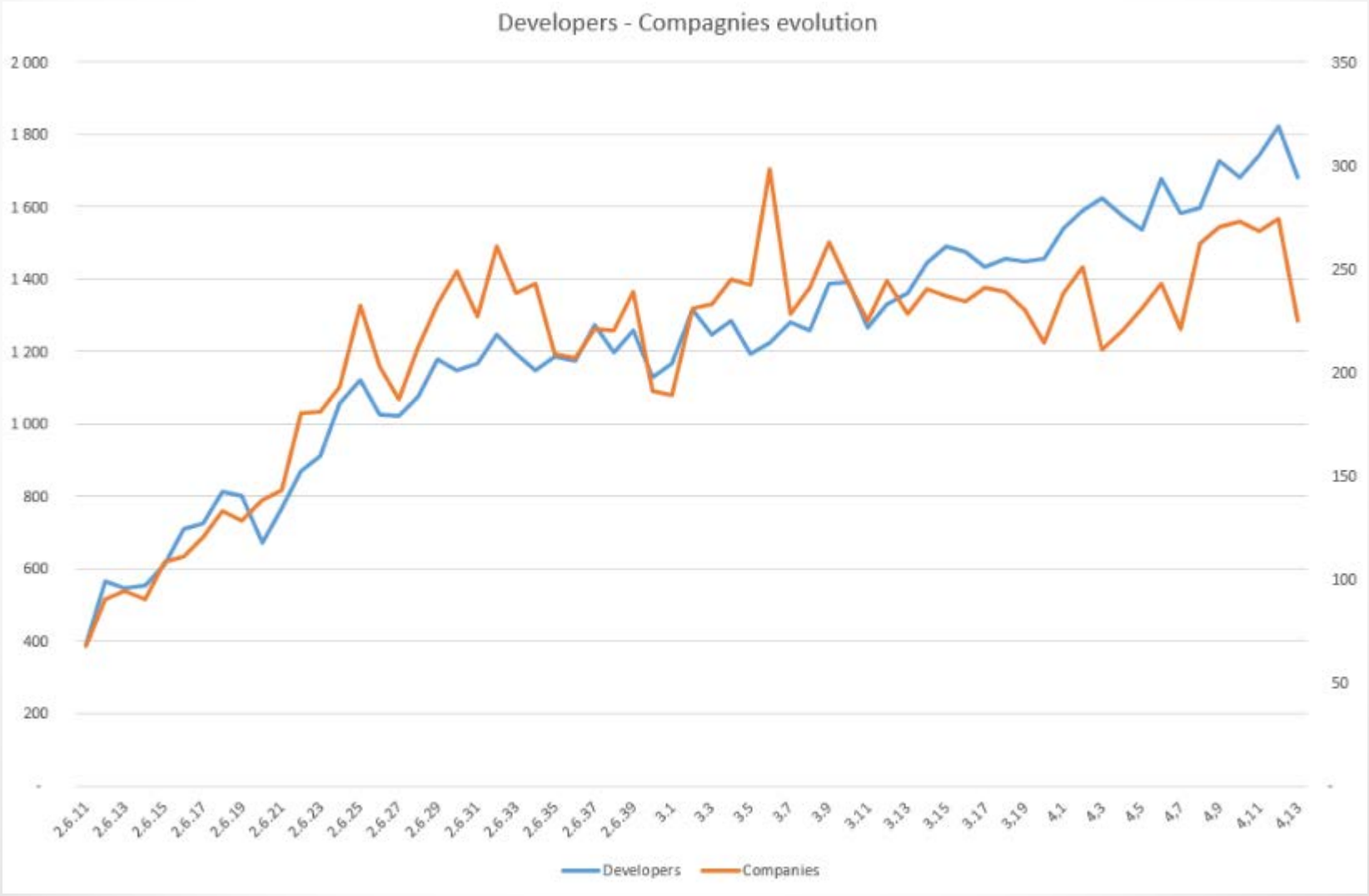
UNIVERSITÉ CÔTE D'AZUR

Windows SLOC

- ✓ Windows NT 4.0 from 1996:11–12 million
- ✓ Windows 2000: more than 29 million
- ✓ Windows XP: 40 - 45 million (sources differ)
- ✓ Windows Vista: 50 million
- ✓ Windows 7: 40 million
  - Reduced probably the crap/bloatware vista had
- ✓ Windows 8: 50 - 60 million
- ✓ Windows 10: 60 without Cortana and 65 with Cortana

UNIVERSITÉ CÔTE D'AZUR

Données Evolutions Linux



UNIVERSITÉ CÔTE D'AZUR

Architectures Matérielles Supportées

- ✓ Regarder dans le répertoire `arch/`
- ✓ Minimum: processeurs 32 bits, avec ou sans MMU
- ✓ Architecture 32 bits
  - alpha, arc, **arm**, blackfin, cris, h8300, hexagon, m32r, m68k, m68knommu, metag, microblaze, mips, nios2, parisc, powerpc, ppc, s390, sh, sparc, um, unicore32, **x86**, xtensa, ...
- ✓ Architecture 64 bits
  - **arm64**, c6x, **ia-64**
- ✓ Voir la documentation dans les sources pour plus de détails

UNIVERSITÉ CÔTE D'AZUR

Construction du Noyau (≥ 2.6)

- ✓ **Configuration de la compilation:**
  - `make config`, `make menuconfig`, `make xconfig` **ou** `make gconfig` **: modifier une configuration existante**
  - `make defconfig`: **configuration par défaut pour l'architecture**
  - `make oldconfig`: **validation et mise à jour d'une ancienne configuration**
  - `make allnoconfig`: **intéressant pour l'embarqué permet d'avoir une configuration minimum du noyau**
    - **environ 640Ko en bzImage**
    - **Inclus les options nécessaires au fur et à mesure des besoins**
- ✓ **Compilation du noyau et des modules**
  - `make` **: tout est compilé (noyau et modules)**
- ✓ **Nettoyage des sources:**
  - `make clean` **: nettoie les sources des fichiers compilés**
  - `make mrproper`: **idem clean + supprime le fichier de config**

UNIVERSITÉ CÔTE D'AZUR

Installation Définitive du Noyau

- ✓ La solution la plus simple (configuration automatique du chargeur en général):
  - `make modules_install ; make install`
- ✓ Ou installation manuelle du noyau sur une cible distante:
  - `cd …/linux-x.y.z-e`
  - `cp System.map /boot/`
  - `cp arch/x86/boot/bzImage /boot/vmlinuz-x.y.z-e`
  - `make modules_install`
    - Copie les modules dans `/lib/modules/x.y.z-e/`
  - Reconfigurer le chargeur de noyau en conséquence
    - Créer des liens `/vmlinuz` et `/initrd`

UNIVERSITÉ CÔTE D'AZUR

Dans quels cas Compiler un Noyau ?

- ✓ Pour un PC standard (du point de vue matériel), pas de réel intérêt
  - Mise à jour régulière de la version du noyau
  - Mise à jour de sécurité suivies dans les distributions majeures
- ✓ Donc dans quels cas est-ce nécessaire de compiler son noyau
  - Développer pour le noyau
  - Activer un pilote de périphérique non prévu, non disponible en module (matériel spécifique)
  - Activer une fonctionnalité qui n'est pas encore dans la branche principale du noyau
    - Ajout du support graphique au boot
    - Ajout du support temps réel (application d'un patch)
    - Ajout d'un driver spécifique (carte tuner TV TNT, satellite, ...)
  - Faire un noyau optimisé dans le cas de l'embarqué bien sûr !

Environnement de Travail

Virtualisation:

Faciliter les Travaux Dirigés...

mais il ne sont pas virtuels pour autant

UNIVERSITÉ CÔTE D'AZUR

Qu'est-ce que la Virtualisation ?

- ✓ **But**
  - Faire tourner plusieurs systèmes, simultanément, sur une seule machine
- ✓ **Notions**
  - Couche d'abstraction matérielle et/ou logicielle
  - Système d'exploitation hôte (host) installé directement sur le matériel
  - Systèmes d'exploitations (ou applications, ou encore ensemble d'applications) « virtualisé(s) » ou « invité(s) » (guest)
  - Partitionnement, isolation et/ou partage des ressources physiques et/ou logiques
  - Images manipulables : démarrage, arrêt, gel, clonage, sauvegarde et restauration, sauvegarde de contexte, migration d'une machine physique à une autre
  - Réseau virtuel : réseau purement logiciel, interne à la machine hôte, entre hôte et/ou invités

UNIVERSITÉ CÔTE D'AZUR

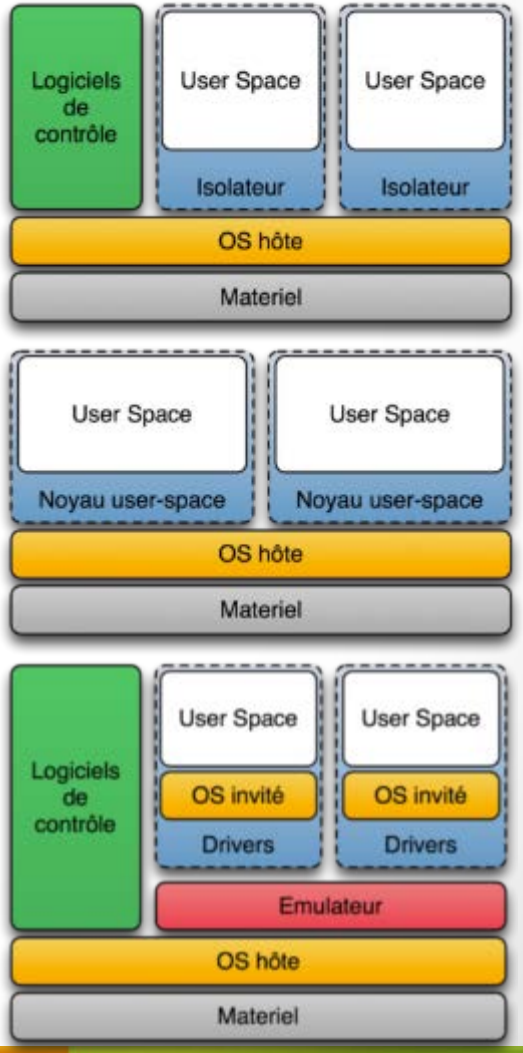
Virtualisation et Embarqué

- ✓ Les principes et différentes approches de la virtualisation sont importantes de nos jours
- ✓ Utilisé dans de nombreux domaines:
  - Votre machine de travail
  - Les ordinateurs du Cloud
  - Mais aussi dans l'embarqué:
    - Système de base « validé » et minimaliste
    - Ajout de Système(s) virtualisé(s) pour des choses moins importantes
    - Exemple dans l'automobile
      - Système validé pour la gestion éléments critiques (ABS, ...)
      - Système GNU/Linux virtualisé pour les aspects multimédia

UNIVERSITÉ CÔTE D'AZUR

Différents Types de Virtualisation 1/2

- ✓ Isolation
  - ✓ Isoler l'exécution des applis dans des contextes d'exécution
  - ✓ Performance mais pas réellement de virtualisation
  - ✓ Ex: chroot, jail, Linux-VServer
- ✓ Noyau en Espace Utilisateur
  - ✓ Noyau fonctionnant en *user space* comme une application
  - ✓ Indépendance par rapport au système hôte inexistante
  - ✓ Ex: User Mode Linux, coLinux
- ✓ Machine Virtuelle
  - ✓ Logiciel qui émule et/ou virtualise le matériel pour les OS *guest*
  - ✓ Isole bien les OS, mais coût en performances
  - ✓ Ex: Qemu, VirtualPC, VirtualBox, VMware



UNIVERSITÉ CÔTE D'AZUR

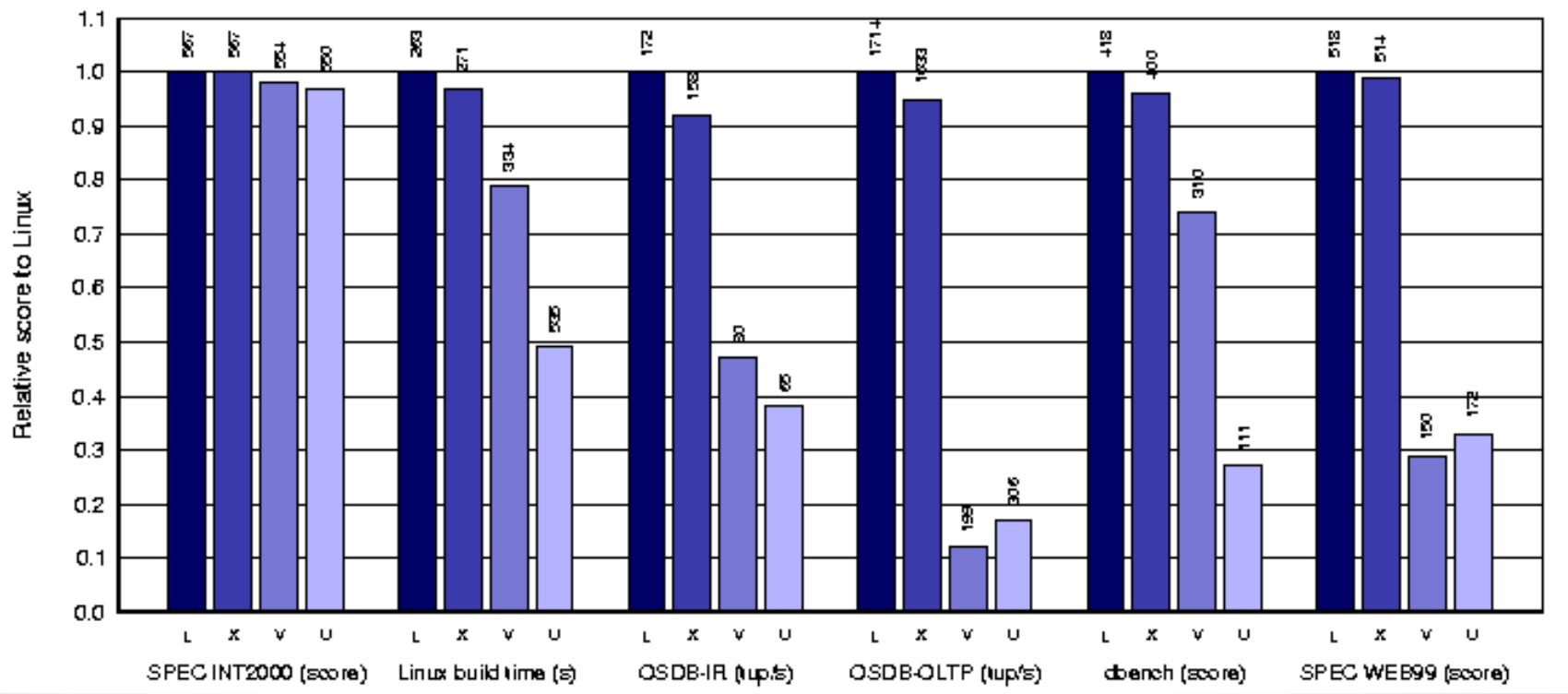
Différents Types de Virtualisation 2/2

- ✓ Emulateur
  - Permet l'exécution d'un programme pour un système X sur un système Y
  - Une instruction est exécutée par une routine qui simule le PC
  - Lent mais universel
  - Ex: Emulateurs PSX, DS, Néo-Géo, … mais aussi Plex86, QEMU
- ✓ Virtualisation complète
  - Compile les instructions lors de leur 1ère exécution (Virtual PC)
  - Si possible, le code est directement exécuté sur le CPU, sinon, ré-écriture dynamique. Permet une exécution à 80% de la vitesse (Vmware, Kqemu, VirtualBox)
- ✓ Para-Virtualisation ou Hyperviseur
  - Le système invité à « conscience » du système sous-jacent
  - Performances optimales (proches d'un système hôte)
  - Ex: Xen, UML, Hypervisor, …

UNIVERSITÉ CÔTE D'AZUR

Performances

✓ Comparaison des performances



Source: http://www.cl.cam.ac.uk/research/srg/netos/xen/performance.html

UNIVERSITÉ CÔTE D'AZUR

Mais quelle VM pour la virtualisation?

- ✓ Plusieurs logiciels sont disponibles:
  - Qemu
    - Emulation et virtualisation, logiciel libre, Fabrice Bellard
  - VMware (Player)
    - Virtualisation, logiciel propriétaire « gratuit », société EMC Corp.
  - VirtualPC
    - Virtualisation, logiciel propriétaire gratuit, société Microsoft
  - VirtualBox (ou Oracle VM VirtualBox)
    - Virtualisation, logiciel libre, société Oracle
- ✓ Nous utiliserons VirtualBox car:
  - Disponible gratuitement sur tous les environnements
  - Très facile d'utilisation et interopérable
    - disque virtuel .vmdk utilisable sous VMware et VirtualBox

UNIVERSITÉ CÔTE D'AZUR

VirtualBox

- ✓ Fournit un environnement virtuel
  - Processeur, Mémoire, Bus, ...
  - Périphériques: Carte réseau, Carte vidéo, ...
- ✓ Machine hôte (host)
  - Machine exécutant le programme VirtualBox
- ✓ Machine virtuelle invitée (guest)
  - Machine s'exécutant à l'intérieur du processus VirtualBox
- ✓ Communications
  - Addons guest pour des fonctionnalités supplémentaires :
    - copier/coller, glisser/déposer
    - mais multi-résolution graphique, partage de dossiers, ...
  - Utilisation de TCP/IP pour communiquer
    - La machine invitée utilise la machine hôte comme passerelle
    - Binding réseau mis en place automatiquement sur l'interface réelle connecté