Université Nice Sophia Antipolis

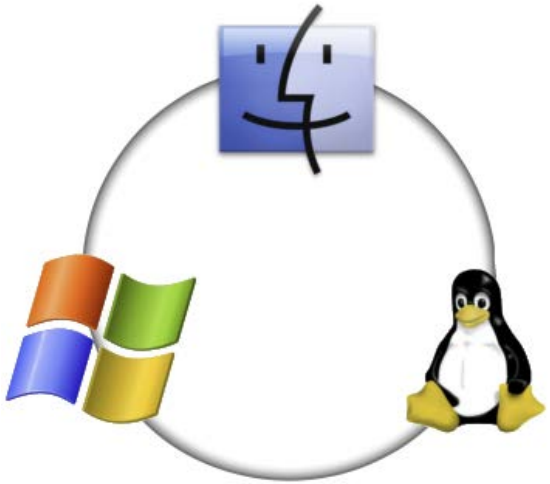
Systèmes d'Exploitation Avancés

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

(*) Cours réalisé grâce aux documents de :
Olivier Dalle, Erick Gallesio, Fabrice Huet, Stéphane Lavirotte, Michael Opdenacker, Jean-Paul Rigault

Mail: Stephane.Lavirotte@unice.fr
Web: http://stephane.lavirotte.com/
Université de Nice - Sophia Antipolis

Université Nice Sophia Antipolis



Systèmes d'Exploitation Avancés

Présentation du Cours

Université Nice Sophia Antipolis

Objectifs du Cours

- ✓ But:
  - Approfondir les connaissances sur les Systèmes d'Exploitation
  - Avoir une vision système « de l'intérieur » au plus proche du matériel
  - Etudier les différentes composantes d'un S.E.
  - Ajouter des fonctionnalités au noyau
- ✓ Axé sur l'expérimentation
  - Illustration et études de cas sur Linux
- ✓ Appliqué aux systèmes embarqués
  - Voir des spécificités pour les Systèmes Embarqués

Université Nice Sophia Antipolis

Organisation du Cours

- ✓ Organisation
  - 8 Cours et TD les mardi matin, évaluations incluses
  - 2 notes au minimum
- ✓ Intervenants
  - Académiques
    - Stéphane Lavirotte (Université de Nice – Sophia Antipolis)
- ✓ Plan du Cours:
  - Introduction, Architecture et Prise en main du noyau (C 2h, TD 2h S. Lavirotte)
  - Ajout de fonctionnalités au noyau: les modules (C 2h, TD 2h, S. Lavirotte)
  - Pilote de périphériques (C 2h, TD 2h, S. Lavirotte)
  - Développement de pilotes logiciel (TD 4h, S. Lavirotte)
  - Développement de pilotes matériel, USB (C 1h, TD 3h)
  - Système de fichiers (C 1h, TD 3h, S. Lavirotte)
  - Ordonnancement et Processus (Cours 2h, TD 2h)
  - Evaluation (2 à 4h à définir)

Université Nice Sophia Antipolis

Ce que vous devez maîtriser

- ✓ Pour bien commencer ce cours, quelques prérequis
- ✓ Tout ce que vous devez maîtriser ou connaître d'Unix
  - Le côté utilisateur d'un système (GNU/Linux)
  - L'arborescence du système de fichiers
  - Tout est fichier
  - Gestion particulière du réseau sous Unix
  - Gérer les paquetages et installer des logiciels tiers
- ✓ Pour compléter certaines connaissances
  - http://stephane.lavirotte.com/teach/cours/isle/

Université Nice Sophia Antipolis

Bibliographie (1/2)

- ✓ **Ouvrages**
  - **P. Ficheux, E. Bénard,** *Linux embarqué. Nouvelle étude de cas - Traite d'OpenEmbedded*, Eyrolles, 2012.
  - **C. Blaess**, *Développement système sous Linux: Ordonnancement multitâche, gestion mémoire, communications, programmation réseau*, Eyrolles, 2011.
  - **Michael Kerrisk**, *The Linux Programming interface: A Linux and UNIX System Programming Handbook*, 2010.
  - **J Corbet, A Rubini, G Kroah-Hartman**, *Linux Device Drivers (3rd Edition)*, O'Reilly, 2005.
  - **D. Bovet** et **M. Cesati**, *Understanding the Linux Kernel (2ème édition)*, O'Reilly 2002.
  - **R. Card**, **E.Dumas** et **F. Mével**, *Programmation Linux 2.0 - API système et fonctionnement du noyau*, Eyrolles 1996.
  - **A. Silberschatz** et **P. B. Galvin**, *Principes des systèmes d'exploitation* (4ème édition), Addison-Wesley 1994.
  - **M. J. Bach**, *Conception du système Unix*, Masson, Prentice Hall, 1993.
- ✓ **Cours**
  - **Linux pour l'embarqué: http://free-electrons.com/**

Université Nice Sophia Antipolis

Bibliographie (2/2)

- ✓ Ressources en ligne
  - http://www.tldp.org/guides.html,
    - **Gerard Beekmans**, *Linux From Scratch*, Nov 2005.
    - **Peter Jay Salzman, Michael Burian, Ori Pomerantz**, *The Linux Kernel Module Programming Guide*, Kernel 2.4 et 2.6, Jul 2004.
    - **Tigran Aivazian**, *Linux Kernel 2.4 Internals*, Aug 2002.
    - **David A. Rusling**, *The Linux Kernel*, Jan 1998.
    - **Michael K. Johnson**, *The Linux Kernel Hackers' Guide*, Aug 1998.
  - http://www.linuxhq.com/lkprogram.html
- ✓ Journaux grand public
  - GNU / Linux Magazine
- ✓ Documentation noyau
  - `/usr/src/linux/Documentation/`...
  - Les sources !!! (grep, (e|c)tags, ...)

Université Nice Sophia Antipolis

Remerciements

- ✓ Cours réalisé à partir des sources suivantes :
  - Cours et exercices « Système Avancé »
    - Erick Gallesio et Jean-Paul Rigault (1999-2004)
    - Stéphane Lavirotte (2004-2008)
  - Cours et exercies « Systèmes et Applications Embarqués »
    - Stéphane Lavirotte (2008-2015)
  - Cours et exercices
    - Michael Opdenacker – Free Electrons
      http://free-electrons.com/

- ✓ Remerciements à:
  - Olivier Dalle
  - Erick Gallesio
  - Fabrice Huet
  - Michael Opdenacker
  - Jean-Paul Rigault

Université Nice Sophia Antipolis

Introduction et Architecture d'un Système d'Exploitation

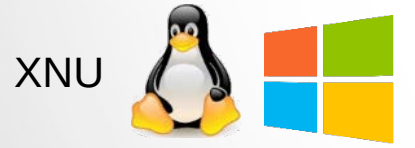
Stéphane Lavirotte

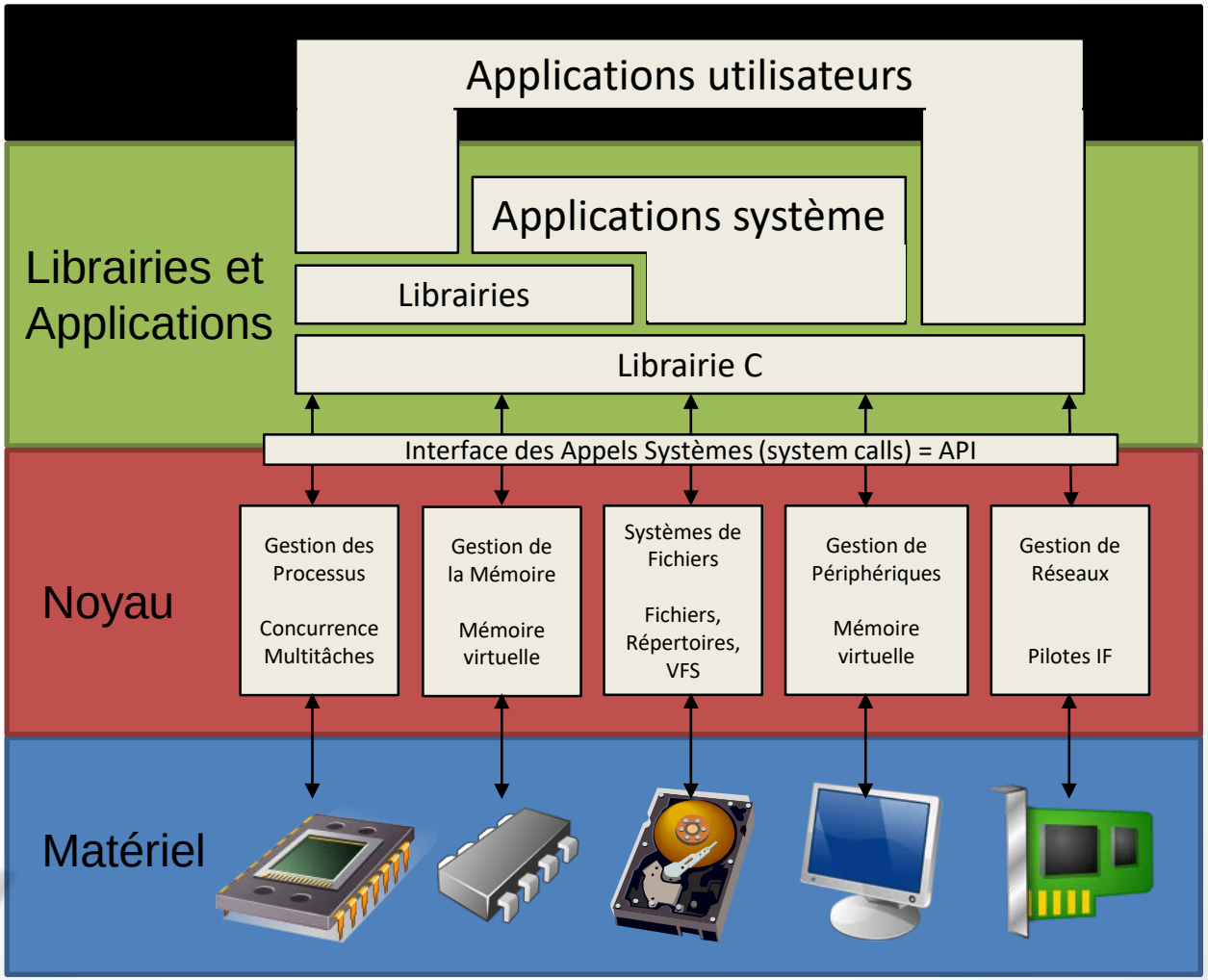
Mail: Stephane.Lavirotte@unice.fr

Web: http://stephane.lavirotte.com/

Université de Nice - Sophia Antipolis

Architecture générale d'un Système d'Exploitation





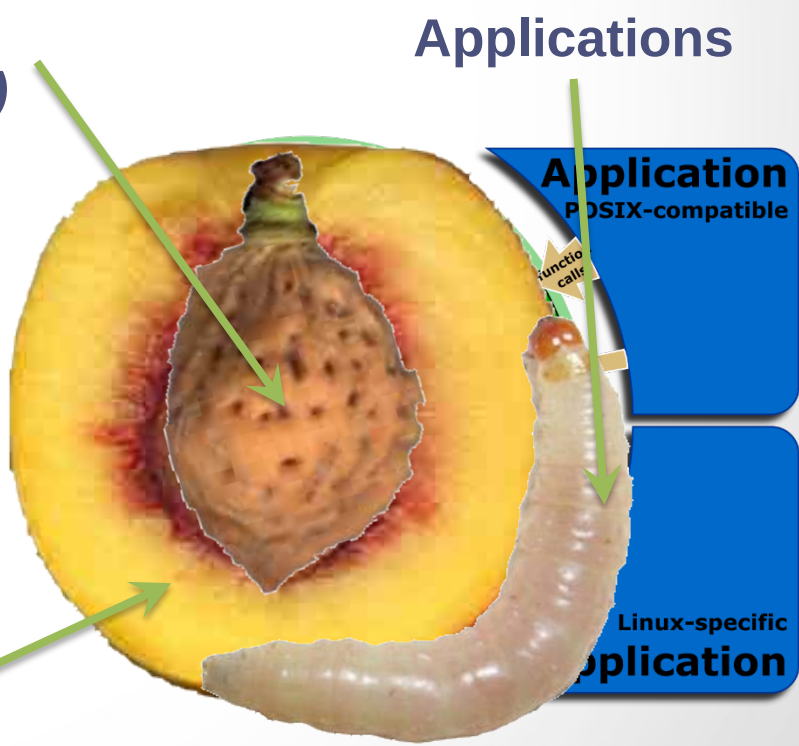
Noyau vs Système d'Exploitation

- ✓ Ne pas confondre:
  - Noyau

  et
  - Système d'exploitation

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

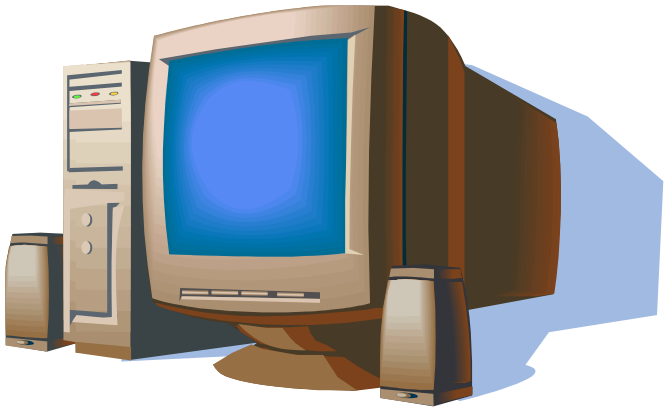
- ✓ Donc on parle de:
  - GNU / Linux



Université Nice Sophia Antipolis

Un Système d'Exploitation: Pour faire quoi?

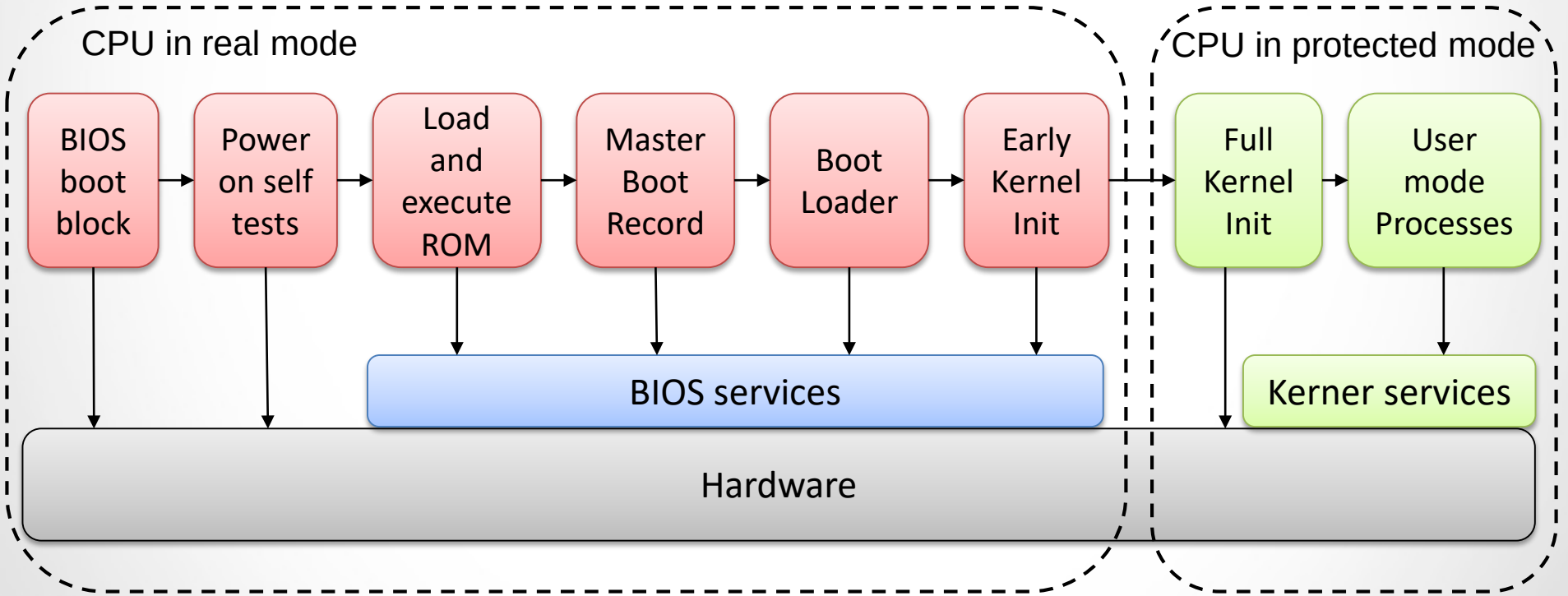
- ✓ Fonctionnalités d'un Noyau
  - Gestion des processus
    - cycle de vie (création, vie, destruction), ordonnancement, ...
  - Gestion de la mémoire
    - allocation, protection, ...
  - Gestion des systèmes de fichiers
    - Formatage, cache d'accès aux fichiers, optimisations ...
  - Gestion de l'accès aux périphériques
    - Bus de communication, échange de données, ...
  - Gestion des requêtes des processus
    - Appels systèmes: lire, écrire, ouvrir, ...
  - Gestion des interruptions et exceptions matérielles
    - Déroutement: division par 0, débordement de pile, ...
    - Interruptions: timer, périphériques, ...
  - Donc fournit une abstraction du matériel, unifier les accès aux périphériques
- ✓ Fonctionnalités du Système d'Exploitation
  - Cycle de vie des services
  - Ensemble de commandes utilisateur pour piloter le système
  - Assurer des tâches d'entretien du système



Démarrer un Système

Il faut bien commencer
par quelque chose

Les Grandes Etapes du Démarrage d'un PC



Boot Loader

- ✓ Programme sur le MBR
  - Le programme du boot sector (MBR) charge le boot loader
  - Le boot loader déclenche le chargement d'un autre OS
    - 1er secteur du Boot Loader installé dans le secteur de boot secondaire (en début de partition Linux)
    - On se retrouve alors dans la même situation, le Boot Loader termine son chargement
- ✓ Le Boot loader: charge le noyau et lance son exécution
  - Options possibles pour paramétrer le noyau

Un chargeur de noyau: GRUB (1/2)

- ✓ GRUB (GRand Unified Bootloader)
  - Le chargeur générique de GNU/FSF
  - Support natif pour de multiples OS (Linux, *BSD, ...)
  - Nombreuses améliorations (« micro shell », ...)
- ✓ Fichier de configuration :
  - `/boot/grub/grub.cfg`
  - Fichier lisible très facilement
  - Définition des différentes possibilités de démarrage (`menuentry`)
  - Passage de paramètres au noyau (`linux`)
- ✓ Identifiant des disques et partitions
  - Utilisation des UUID (nécessite un fichier `initrd` avec des scripts)
  - Ou utilisation de la dénomination `/dev/sda1` (depuis Grub 1.9x)

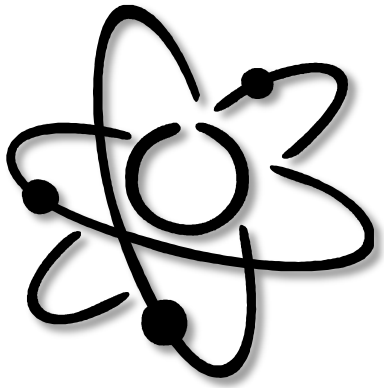
Université Nice Sophia Antipolis

Paramétrage du Noyau au Démarrage

- ✓ Les paramètres sont passés par le bootloader
- ✓ Liste des paramètres du noyau dans le fichier
  - `/usr/src/linux/Documentation/kernel-parameters.txt`
- ✓ Exemple de paramètres:
  - `hdc=ide-scsi` : définit que le périphérique IDE hdc doit être traité comme un périphérique SCSI (pour cdrecord sur un graveur)
  - `init` : donne l'exécutable à lancer après avoir monté la racine
  - `root` : donne la partition racine à monter
  - `ro(rw)` : définit que la partition racine doit être montée en lecture (ou en lecture écriture)
  - `vga` : donne le mode vidéo à configurer
  - `quiet`: ne pas afficher les messages de démarrage du noyau

Chargement du Noyau

- ✓ Le noyau est généralement compressé!
  - Les 1ères instructions = programme de décompression
    - Décompresse le reste du noyau, puis lance l'exécution
- ✓ Le noyau commence son exécution
  - Initialise le noyau, détecte des périphériques, ...
  - Monte la racine en read-only
  - Lance le processus `/sbin/init` (processus 1)
- ✓ Le processus init
  - Lit le fichier `/etc/inittab`
  - Exécute les scripts rc (Run Control)
- ✓ Processus de démarrage des Service sur la machine:
  - Voir System V init ou « SysVinit »



Construire un Noyau

... en passant par l'organisation des sources
et la compilation d'un noyau...

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

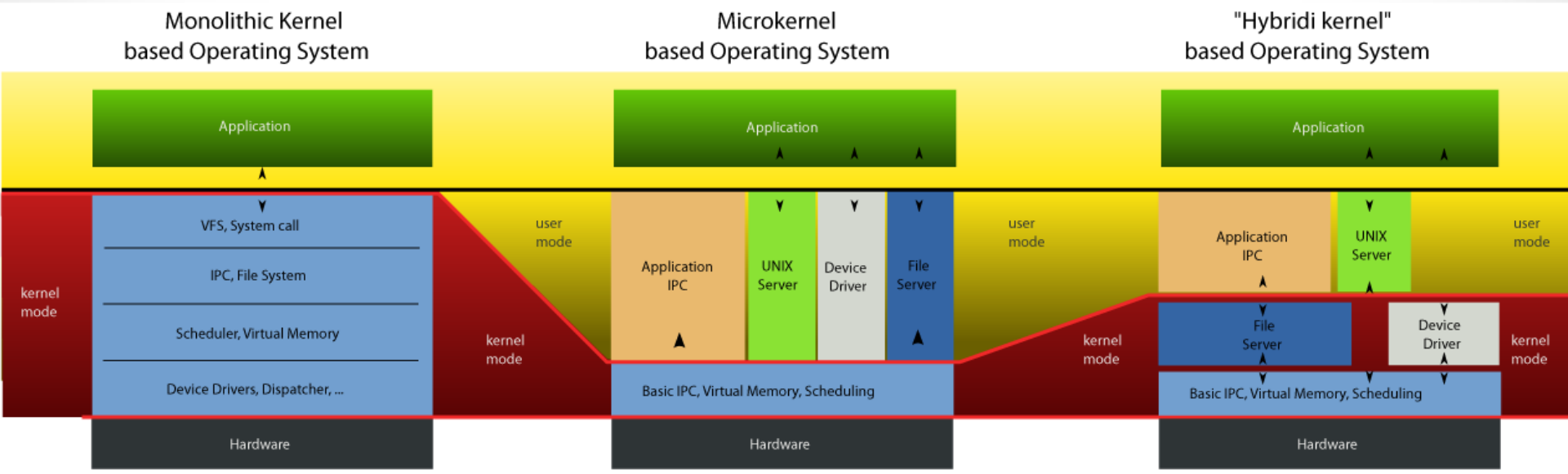
Université Nice Sophia Antipolis

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, OS/2, AIX, Irix, ...
- ✓ Micro Noyau
  - Fournit les services minimaux
    - gestion des processus, de la mémoire et des IPCs
  - Les autres services sont fournis par des progs utilisateurs
  - Ex.: Mac OS X (Mach), GNU/Hurd (Hurd)
- ✓ Hybride
  - Combinaison du meilleur des deux mondes ? (Windows NT)
- ✓ http://fr.wikipedia.org/wiki/Noyau_de_système_d'exploitation

Université Nice Sophia Antipolis

Différents types de Noyau (2/2)



Comparaison des Types de Noyau

- ✓ Noyau Monolithique
  - Plus facile à écrire
  - Moins élégant qu'un micro noyau
  - Plus performant
- ✓ 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

Un exemple de Noyau : Linux

Illustration avec le

2.0 ≤ Noyau Linux ≤ 4.x

Université Nice Sophia Antipolis

Architectures Matérielles Supportées

- ✓ Regarder dans le répertoire `arch/`
- ✓ Minimum: processeurs 32 bits, avec ou sans MMU
- ✓ Architecture 32 bits
  - alpha, arm, arm26, cris, h8300, i386, m32r, m68k; m68knommu, mips, parisc, powerpc, ppc, s390, sh, sparc, um, v850, xtensa
- ✓ Architecture 64 bits
  - ls64, sh64, sparc64, x86_64
- ✓ Voir la documentation dans les sources pour plus de détails

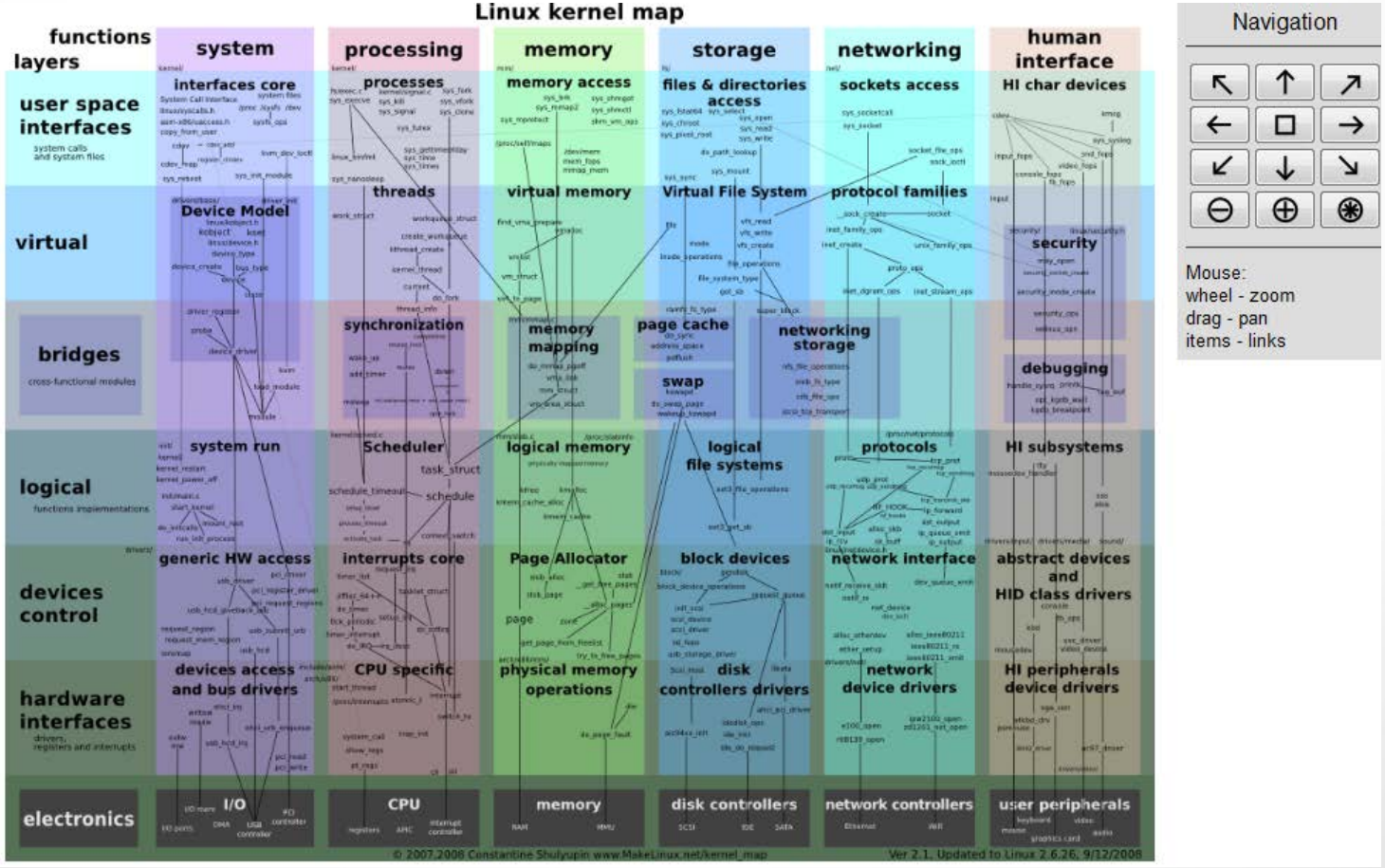
Université Nice Sophia Antipolis

Organisation des Sources Linux

```
/usr/src/linux
|-- Documentation
|-- arch
|   `-- alpha, arm, i386, ia64, m68k, mips, ppc, sparc, ...
|-- drivers
|   `-- block, bluetooth, cdrom, char, ieee1394, net, pci, scsi, ...
|-- fs
|   `-- autofs, ext2, ext3, fat, isofs, msdos, nfs, proc, vfat, ...
|-- include
|   |-- asm -> asm-i386
|   |-- asm-generic
|   |-- asm-i386
|   |-- linux
|   |   `-- modules
|   |-- net
|   |-- pcmcia
|   |-- scsi
|   `-- video
|-- init
|-- ipc
|-- kernel
|-- lib
|-- mm
|-- net
`-- scripts
```

Cartographie des Sources

- ✓ Carte interactive des sources du noyau Linux
  - http://www.makelinux.net/kernel_map/



Université Nice Sophia Antipolis

Construction du Noyau (≥ 2.6)

- ✓ **Nettoyage des sources:**
  - `make clean` : nettoie les sources des fichiers compilés
  - `make mrproper`: idem clean + supprime le fichier de config
- ✓ **Configuration de la compilation:**
  - `make allnoconfig`: intéressant pour l'embarqué permet d'avoir une configuration minimum du noyau
    - environ 550Ko en bzImage
    - Inclus les options nécessaires au fur et à mesure des besoins
  - `make config`, `make menuconfig` ou `make xconfig`: modifier une configuration existante
- ✓ **Compilation du noyau et des modules**
  - `make` : tout est compilé (noyau et modules)

Université Nice Sophia Antipolis

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:
  - `cd /usr/src/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 Nice Sophia Antipolis

Dans quels cas Compiler un Noyau ?

- ✓ Pour un PC standard (du point de vue matériel), pas de réel intérêt de ne pas utiliser le noyau de la distribution
  - 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 ?
  - 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 pilote spécifique (carte tuner TV TNT, ...)
  - Faire un noyau optimisé dans le cas de l'embarqué bien sûr !

Un système qui fonctionne

... et si on commençait à regarder à la loupe ce qui se passe à l'intérieur

Mode Utilisateur et Mode Noyau Définitions

- ✓ **Mode utilisateur**: mode « normal » ou protégé
  - Mode d'exécution pour les programmes utilisateurs
  - Seules les zones mémoires allouées sont accessibles
  - Certaines instructions du processeur sont interdites
  - Accès direct aux périphériques interdit

- ✓ **Mode noyau**: mode privilégié ou superviseur
  - Mode d'exécution du programme noyau
  - Tout l'espace mémoire est adressable
  - Toutes les instructions du processeur sont autorisées
  - Accès direct aux périphériques

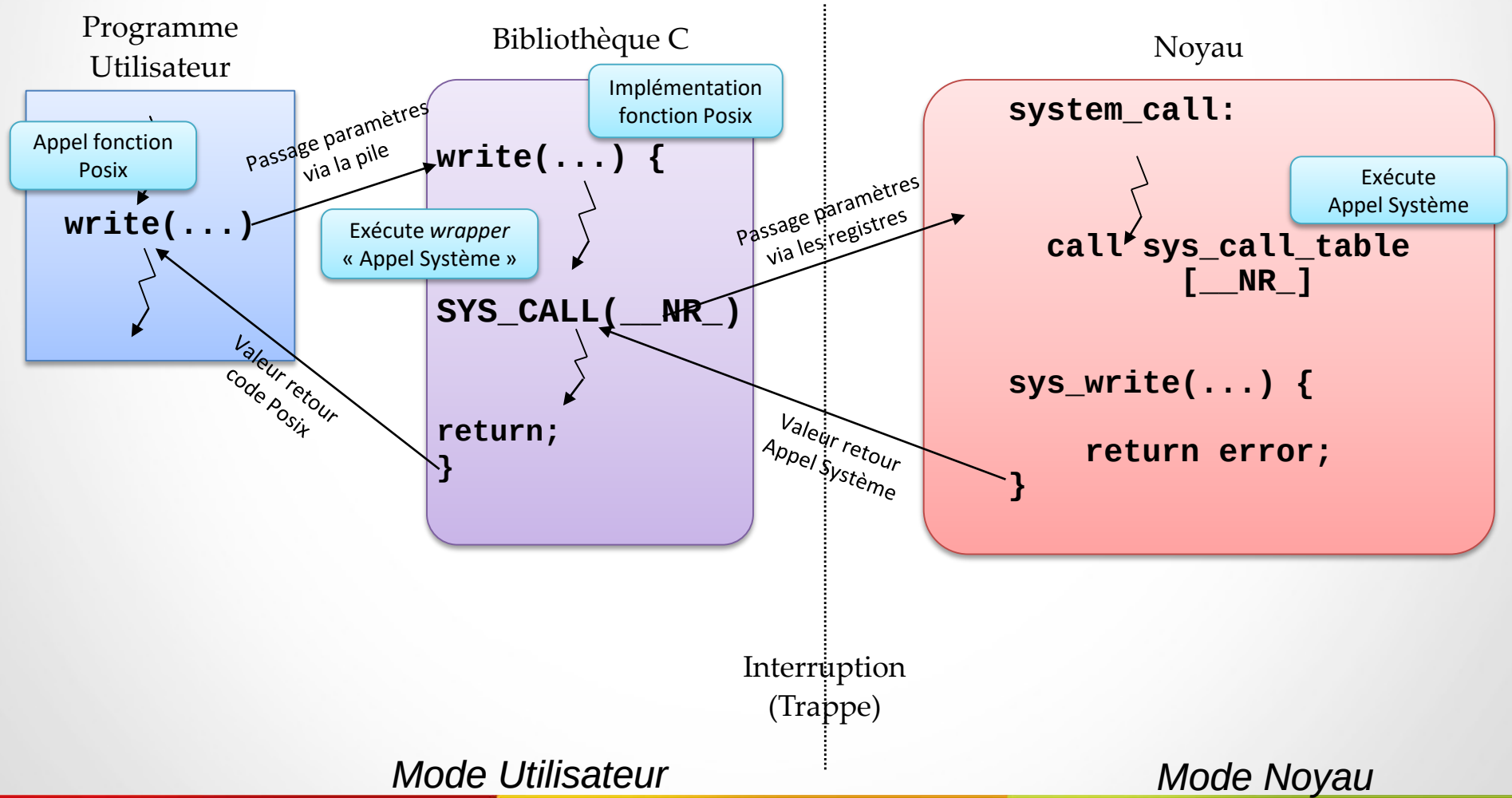
Université Nice Sophia Antipolis

Qu'est ce qu'un Appel Système ?

- ✓ L'ensemble des appels systèmes
  - Est fixe (pas possible d'en ajouter pour un noyau en cours d'exécution)

- ✓ Un appel système
  - Provoque le passage d'un processus du **mode utilisateur** en **mode noyau**
  - Prend un ensemble d'arguments qui spécifie les informations à transférer de l'espace utilisateur à l'espace noyau
  - Est identifié par un numéro unique qui l'identifie
    - La numérotation n'est pas visible par les processus qui identifient un appel système par son nom
  - D'un point de vue programmation = invoquer une fonction C

Illustration du Fonctionnement d'un Appel Système



Université Nice Sophia Antipolis

Détail du Fonctionnement d'un Appel Système

Sous Linux-x86, le passage en mode noyau est implémenté de la façon suivante:

- ✓ Placement des paramètres dans les « bons » registres
  - `eax` pour l'appel système et registres suivants pour paramètres
- ✓ Interruption logicielle `0x80` (trappe)
  - Passage du processus en mode noyau
  - Exécution de la fonction `system_call` (cf `arch/i386/kernel/entry.S`)
    - Récupération du numéro d'appel système dans registre `eax` et des arguments dans les registres suivants (`ebx`, `ecx`, ...)
    - Exécution de la fonction située dans `sys_call_table[eax]`
  - Retour de la fonction `system_call`
  - Retour du processus en mode utilisateur
- ✓ Continuation du programme

Déclaration d'un Appel Système sous Linux

On veut implémenter l'appel système

```
addition(int x, int y)
```

qui affiche le résultat de l'addition de `x` et de `y` sur la console.

Chaque appel système a un nom défini dans la table

`sys_call_table`

(cf `arch/x86/kernel/syscall_table_XX.S`)

Ajouter le nouvel appel système dans la table `sys_call_table`:

```
ENTRY(sys_call_table)
    .long sys_restart_syscall /* 0 - old "setup()" system call */
    .long sys_exit)
    .long sys_fork
    .long sys_read  /* 3 */
    ...
    .long sys_addition) /* Appel ajouté: N° entrée */
```

Ajouter un numéro unique `arch/x86/include/asm/unistd_XX.h`

(cf `<asm/unistd.h>`)

```
#define __NR_addition
```

Université Nice Sophia Antipolis

Implémentation d'un Appel Système sous Linux

- ✓ Ecriture de la fonction implémentant l'appel système (cf `kernel/sys.c`)

```
SYSCALL_DEFINE2(addition, int, x, int, y)
{
    printk(" <1>Addition %d %d ==> %d\n", x, y, x+y);
    return 0;
}
```

- ✓ Remarques:
  - `SYSCALL_DEFINE2` est une macro qui permet de définir un syscall avec 2 arguments
    - Différentes macro pour 1, 2, 3, ... arguments
  - `printk` est le pendant de `printf` de la bibliothèque C (mêmes conventions mais pas de flottants).
  - Messages affichés sur la console et dans le fichier `/var/log/messages`

Tester votre Appel Système sous Linux à partir de 2.6.18

- ✓ **Définition de l'appel système dans la bibliothèque C**

```
#include <asm/unistd.h>  /* inclus the declared system calls */
   #include <sys/syscall.h>
   #include <errno.h>
   #include <stdio.h>

   void main() {
      printf("return call code: %d\n", syscall(__NR_addition, 1, 2));
   }
```

- ✓ **Pour compiler**
  - **Spécifier où sont localisées les fichiers include modifiés**

```
gcc -o test_syscall test_syscall.c -I...
```

- ✓ **Typiquement c'est que qui est utilisé dans la bibliothèque C**

Université Nice Sophia Antipolis

Espace d'adressage Utilisateur vs Noyau (1/2)

- ✓ Distinction entre la mémoire utilisée par le noyau et celle utilisée par les processus
- ✓Le noyau fournit quelques fonctions/macros utilitaires (cf `<asm/uaccess.h>`)
- ✓`int access_ok(int type, unsigned long addr, unsigned long size);`
  - `addr`: adresse dans l'espace d'adressage du processus appelant
  - type ∈ {*VERIFY_READ, VERIFY_WRITE*}
- ✓`int get_user(lvalue, address);`
  - pour lire dans l'espace d'adresse de l'appelant (0 si OK -EFAULT sinon).
  - le type du pointeur "address" détermine la taille à transférer
- ✓`int put_user(expression, address);`
  - pour écrire dans l'espace d'adressage de l'appelant
  - mêmes conventions que get_user
- ✓`__get_user(lvalue, address);` et `__put_user(expression, address);`
  - version "unsecure" (i.e. ne font pas un test avec access_ok)

Espace d'adressage Utilisateur vs Noyau (2/2)

- ✓ `int copy_to_user(unsigner long to, unsigned long from, unsigned long size);`
  - pour copier une zone mémoire depuis l'espace d'adressage du noyau vers l'espace d'adressage du processus.
  - valeur de retour = nombre d'octets non transférés
- ✓ `int copy_from_user(unsigned long to, unsigned long from, unsigned long size);`
  - pour copier une zone mémoire depuis l'espace d'adressage du processus vers l'espace d'adressage du noyau.
  - mêmes conventions
- ✓ `int __copy_to_user(...)` et `__copy_from_user(...)`
  - version "unsecure" (i.e. ne font pas un test avec access_ok)

Virtualisation pour les TDs

Faciliter les Travaux Dirigés...

mais il ne sont pas virtuels pour autant

Université Nice Sophia Antipolis

Utilisation de la Virtualisation

- ✓ But
  - Faire tourner plusieurs systèmes, simultanément, sur une seule machine
- ✓ De nombreux avantages:
  - ✓ Evite les redémarrage de la machine physique
  - ✓ Isole la machine de test
  - ✓ Peu de perte de performance par rapport à la machine native
- ✓ Utilisation de VirtualBox (ou Oracle VM VirtualBox)
  - Virtualisation, logiciel libre, société Oracle
  - Disponible gratuitement sur tous les environnements
  - Très facile d'utilisation et interopérable

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 invité » pour des fonctionnalités supplémentaires :
    - copier/coller, glisser/déposer
    - 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

Addons Invités

- ✓ Mettre à jour les addons invités
  - Si changement de version de VirtualBox
  - Si changement de version du noyau
    - Addons = modules
- ✓ Donc réinstaller les addons invités si perte des fonctionnalités avancées
  - copier/coller, multi-résolution, ...

- ✓ Pour mettre à jour les addons invités sur la VM distribuée
  - Périphérique / Insérer l'image CD...
  - `mount /dev/cdrom /media/cdrom`
  - `/media/cdrom/VMVBoxLinuxAdditions.run`
  - `umount /media/cdrom`

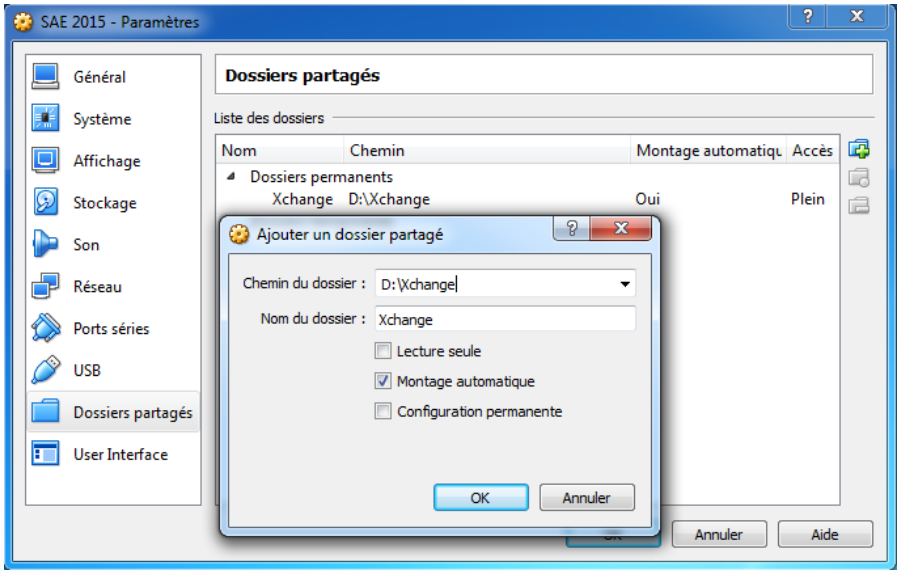
Multi-résolution

- ✓ Pour modifier la résolution de l'écran
  - Démarrer l'environnement graphique
    - `startx`
  - Tester les résolutions disponibles
    - `xrandr`
  - Modifier la résolution pour une autre
    - `xrandr --size 1024x768`
  - Rendre cette modification pérenne au prochain redémarrage
    - Modifier le fichier `/etc/X11/xorg.conf`

```
Section "Screen"
    SubSection "Display"
        Modes "1024x768"
```

Echange de Fichiers entre Hôte et Invité et vice versa

- ✓ Configuration de la machine virtuelle dans VirtualBox:
  - VirtualBox / Configuration / Dossiers partagés
  - Sélectionner le(s) dossiers de la machine hôte à partager
  - Sélectionner le montage automatique



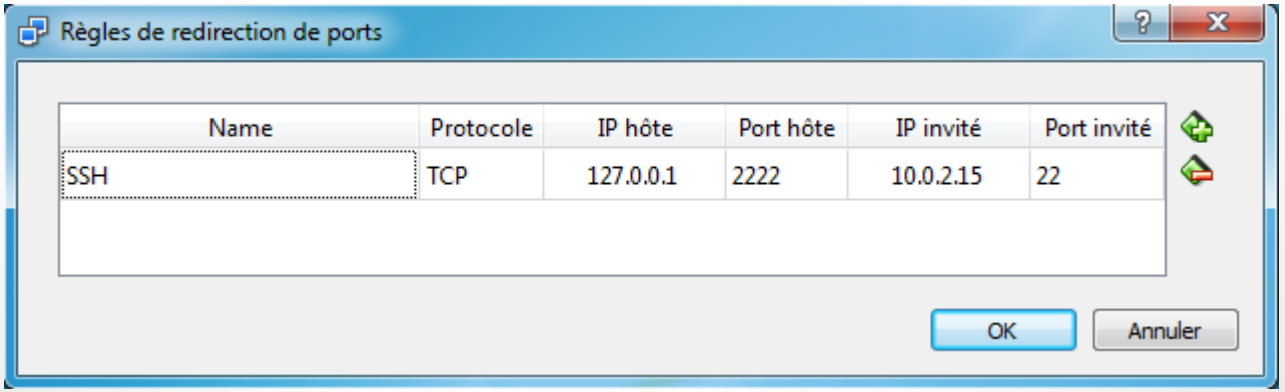
- ✓ Votre dossier sera accessible dans la machine invitée
  - Sous le dossier /media/sf_*Nom*

Partage de fichiers entre Hôte et Invités

- ✓ Les propriétés et gestion des droits sont ceux du système natif qui partage le dossier
- ✓ Donc si partage dossier entre hôte Windows et invité Linux
  - Pas de possibilité de gérer les droits
  - Pas de possibilité de créer des liens (symboliques ou physiques)

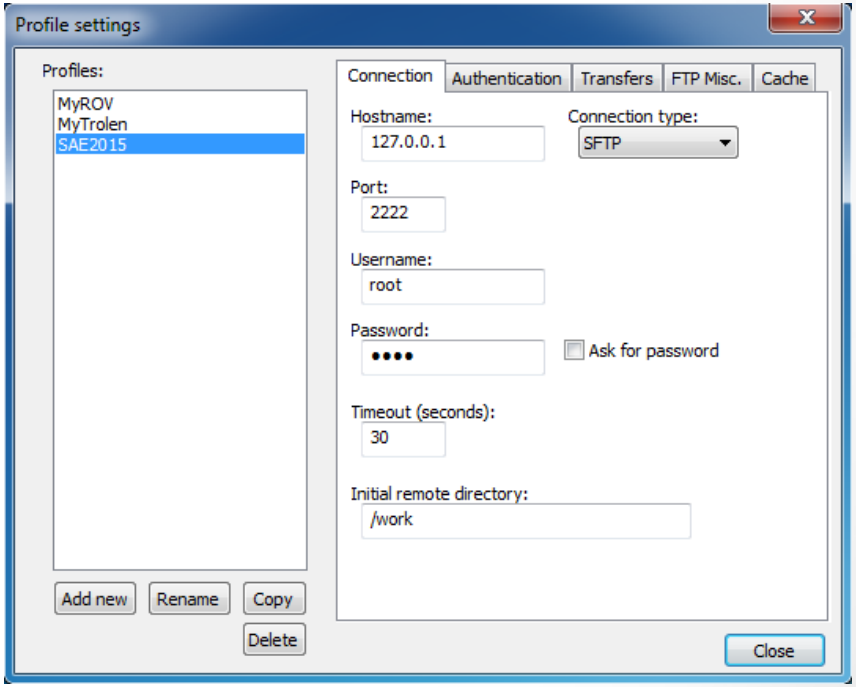
Utiliser un Editeur depuis l'Hôte 1/2

- ✓ Limitation des éditions sur la machine invitée
  - Plus de confort sur la machine hôte et les outils « habituels »
- ✓ Installation de SSH et forwarding de port:
  - Configurer un serveur ssh sur l'invité
    - sudo apt-get install openssh-server
  - Configurer le forwarding de port Hôte – Invité
    - VirtualBox / Configuration / Réseau / Redirection de ports



Utiliser un Editeur depuis l'Hôte 2/2

- ✓ Exemple: Utiliser Notepad + NppFTP
  - Configuration du profile



- ✓ Lors de la sauvegarde, le fichier sera poussé sur la machine invitée par ssh

Commandes Utiles

- ✓ `mount`
- ✓ `poweroff`