Master 1 Informatique

Approfondissement Système

Fabrice Huet
(d’après le cours d’Olivier Dalle)

1. Introduction

Organisation du Cours

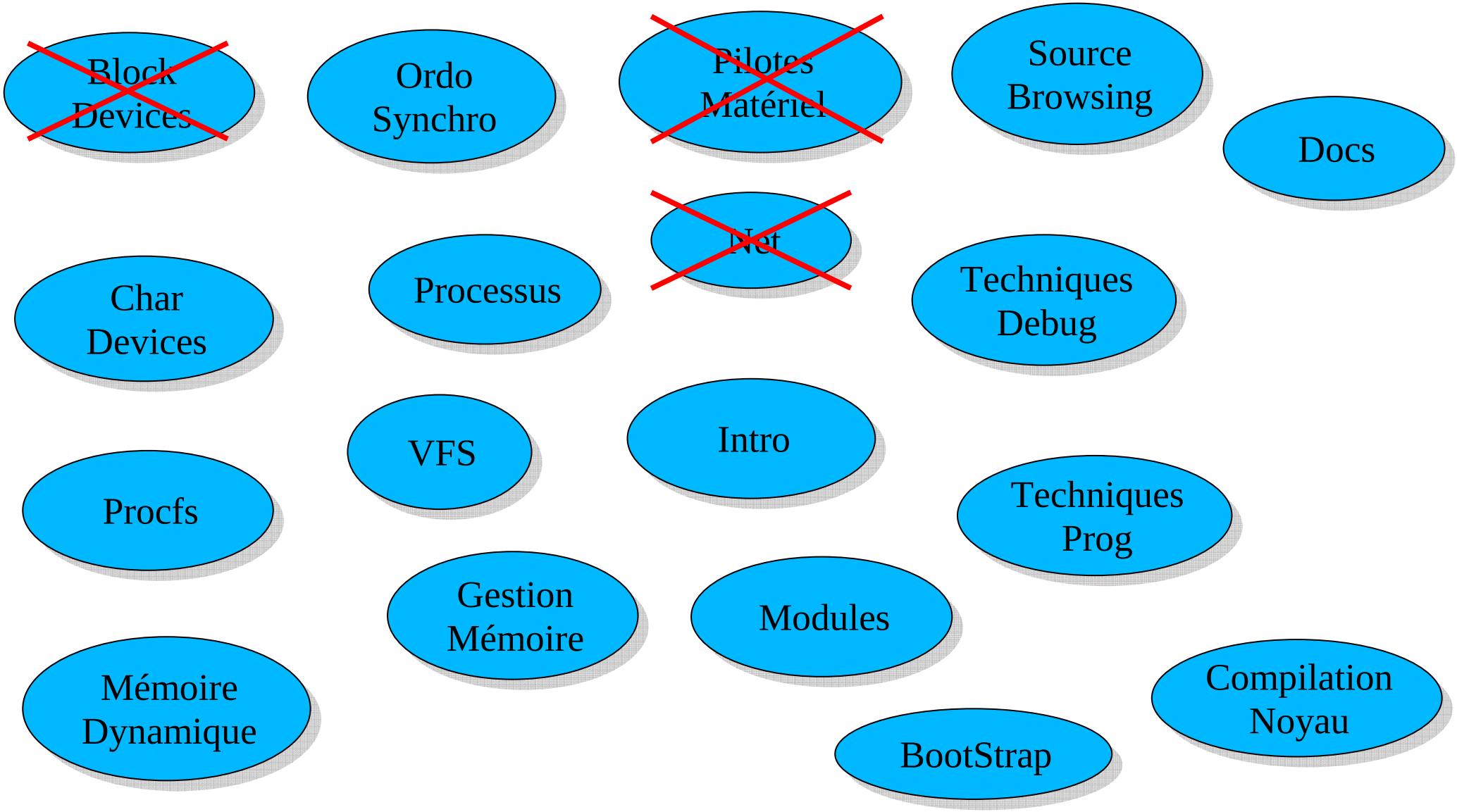
- 12 x 2h Cours le jeudi matin (F. Huet)
- 12 x 2h TP le vendredi matin (G. Chazarain)
  - **Utilisation de QEMU + Linux RH**
- Contrôle des Connaissances :
  - Examen : 70%
  - Projet à rendre

Objectifs du Cours

- Un niveau de plus dans la compréhension d'un S.E.
  1. Utilisation d'applications (shell, outils)
  2. Programmation d'applications (langage C, prog système)
  3. *Gestion, Administration Système*
  4. ***Programmation Noyau***
  5. *Recherche ...*

- *Etude des différents composants d'un S.E.*
  - *Comparaison de différentes solutions*
  - *Illustration et études de cas avec Linux*
    - *Analyse des sources du noyau Linux*
    - *Programmation d'extensions « amusantes »*

Contenu du Cours



Références

- Livres « Compagnons » de O'Reilly
  - **Understanding the Linux Kernel** (Bovet&Cesati, 2e ed)
    Décrit les « Internals » du Noyau 2.4
  - **Linux Devices Drivers** (Rubini&Corbet, 2e ed)
    Orienté programmation (mais néglige « internals »...)
- ***Unix Internals : the New Frontier*** *(Vahallia, Prentice-Hall)*
  Compare implémentations de plusieurs OS : Mach, Solaris, BSD 4.4, ...
- ***Modern Operating Systems*** *(Tanenbaum, Prentice-Hall)*
- *Autres ouvrages intéressants*
  - ***Linux Core Kernel Commentary*** *(Maxwell, Coriolis)*
  - ***Linux Kernel Internals*** *(Beck, Addison-Wesley)*
  - *+ Classiques : Bach (SysV), McKusick (BSD 4.4), ...*

Autres Ressources

- Ressources du Web
  - http://www.linuxhq.com/lkprogram.html
    Point de départ très utile, nombreux liens
  - www.google.com ...
- La documentation du noyau
  - /usr/src/linux/Documentation
  - Génération DocBook des sources du noyau
    - Cibles Makefile principal :
      - make psdocs pdfdocs htmldocs
- Les sources !!
  - Génération de tags : commandes ctags/etags ...

Génération de Tags

- Tags : navigation dans les sources (emacs, vi)
  - Exemple dans emacs :
    - <meta>+<.> (point) sur symbole :
      - recherche définition du symbole
      - ouvre fichier source et se positionne sur la définition trouvée
    - <meta>+<,> (virgule) : passe à la définition suivante
  - Préparation : Indexation à l'aide d'une commande externe comme etags (ctags)
    - Eviter la génération brutale (etags */*.[hc] */*/*.[hc] ...) !
      - Globbing dépasse nb arguments maxi d'une ligne de commande
      - Ordre de parcours pas très intéressant
      - Parties redondantes inutiles (architecture dépendantes)

Exemple de Makefile pour Générer des Tags

```
LINUX_DIR=/usr/src/linux
INCLUDE_DIRS= $(LINUX_DIR)/include/linux $(LINUX_DIR)/include/linux/* \
              $(LINUX_DIR)/include/asm

SRC_CORE_DIRS=  $(LINUX_DIR)/kernel $(LINUX_DIR)/mm $(LINUX_DIR)/init \
                $(LINUX_DIR)/fs $(LINUX_DIR)/net

SRC_DRIVERS_DIRS=       $(LINUX_DIR)/drivers/* $(LINUX_DIR)/drivers/*/*

SRC_FS_DIRS= $(LINUX_DIR)/fs/* $(LINUX_DIR)/fs/*/*

SRC_NET_DIRS= $(LINUX_DIR)/net/core $(LINUX_DIR)/net/ipv4 \
              $(LINUX_DIR)/net/unix

etags:
      etags -I $(INCLUDE_DIRS:=/*.h) $(SRC_CORE_DIRS:=/*.c) -o ktags
      etags -I -a $(SRC_DRIVERS_DIRS:=/*.c) -o ktags
      etags -I -a $(SRC_FS_DIRS:=/*.c) -o ktags
      etags -I -a $(SRC_NET_DIRS:=/*.c) -o ktags

clean:
      rm -f *~ ktags
```

Tour du Propriétaire

- Machine virtuelle QEMU
  - Emulation d'un PC et de son BIOS
- Pour les TP
  - RedHat Centos 3.8
  - Noyau Linux 2.4.X
  - Login `root ou tplinux`
    - pwd : `tplinux`
  - Sources linux installés et recompilés dans /usr/src

2. Système et Architecture

Quelles fonctionalités fournit le système ?

- Principalement 4 choses :
  1) Servir les requêtes explicites des processus
     - Appels systèmes : open(2), read(2), date(2), ...
  2) Traîter les exceptions matérielles dues aux processus
     - Déroutements : Division par 0, débordement de pile, ...
  3) Interruptions matérielles provoquées par les périphériques
     - Interruption disque, réseau, souris/clavier, ...
  4) Un ensemble de processus de service spéciaux
     - Assurent des tâches globales « d'entretien »
       - Swapper
       - Pagedaemon ...

Qu'est-ce que le noyau ?

- Le noyau est un programme spécial
  - S'exécute directement « sur » le matériel
  - A un accès privilégié au matériel
  - Implémente l'abstraction processus
    - Mais n'est pas lui-même un processus...
    - Son rôle est
      - de gérer les processus (création, arbitrage, ...)
      - de leur fournir des services (accès au matériel, ...)
  - Programme résident sur disque
    - Par exemple dans `/vmlinuz` ou `/boot/vmlinuz`
    - Chargé lors de la séquence de démarrage (*bootstrap*)

Que fait le noyau ?

- *Une fois chargé, le noyau :*
  - *Initialise le système*
  - *Crée les processus initiaux (en particulier init)*
    - *Créent à leur tour de nouveaux processus ...*
  - *Reste en mémoire (jusqu'à l'arrêt du système)*

Gestion de la mémoire - 1

- Unix s'appuie sur de la mémoire virtuelle
  - Les adresses mémoires dans les programmes ne font pas directement référence à la mémoire physique
    - 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

Gestion de la mémoire - 2

- Utilisation de mémoire virtuelle avec 2 autres techniques
- Pagination
  - La mémoire est divisée en morceaux
  - L'allocation se fait par nombre entier de pages
  - Peu de fragmentation externe (espace libre coupé en petits morceaux)
  - Sujet à fragmentation interne (surallocation)
  - Nécessite une MMU en hardware

Gestion de la mémoire - 3

- Segmentation
  - Permet d'isoler des morceaux de mémoire
  - Ils ne sont pas accessibles par le processus en cours d'exécution
  - Utilisation de registres hardware
  - Si non respect, *segmentation fault*
- Ces deux techniques permettent d'assurer une protection de la mémoire
  - Un processus ne peut lire/écrire dans la mémoire d'un autre

Kernel Space/User Space

- Distinction entre la mémoire utilisée par le noyau et celle utilisée par les processus
- *Kernel Space*
  - zone mémoire d'exécution du noyau
  - Jamais mis dans le swap
- *User Space*
  - zone mémoire d'exécution des programmes utilisateurs
- Pour accéder au *kernel space* un programme utilisateur doit passer par des appels système.

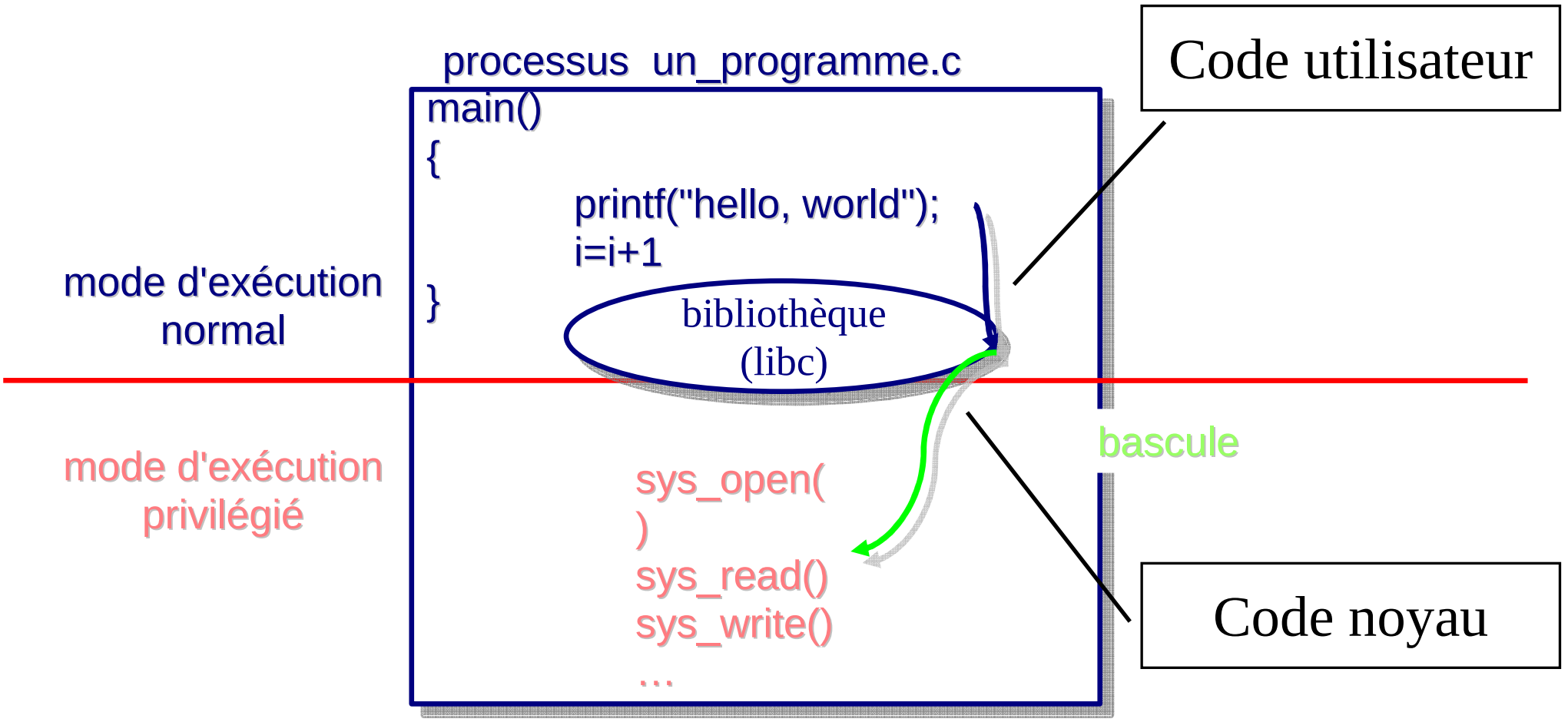
Modes d'Exécution (du processeur)

- 2 modes d'exécution sont indispensables pour Unix
  - **Mode Utilisateur** (normal)
    - Mode d'exécution pour les programmes utilisateurs
    - Protection forte contre les erreurs de programmation
  - **Mode Noyau** (privilégié)
    - Mode d'exécution du programme noyau
    - Instructions machine supplémentaires
    - Registres supplémentaires
      - Gestion de la mémoire virtuelle
    - Accès à l'*espace mémoire utilisateur*
      - *Mémoire (virtuelle) du processus courant*
    - Accès à l'*espace mémoire système*
- Utilisation des *rings* 3 et 1 des processeurs x86

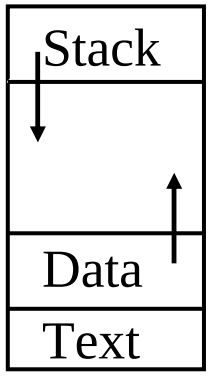
Appels Système

- Un système mono CPU ne peut exécuter qu'une instruction à la fois
- Un processus ayant besoin de services de l'OS doit donc lui passer le contrôle
- L'OS regarde la demande du processus
- L'éxécute
- Rend la main au processus

Illustration : Passage en Mode Noyau



Mémoire et pile



- La mémoire d'un processus est divisée en 3 zones
  - Pile : croissance automatique
  - Data : croissance explicite
  - Text : fixe

- Utilisée par la plupart des langages
  - En particulier le langage C
    - Donc le noyau
  - Quelques exceptions : Fortran77, ...
- Chaque processus en possède (au moins) une
  - Elle permet de mémoriser les paramètres d'appel des fonctions
    - Autorise les appels récursifs
  - Lorsque l'exécution de la fonction se termine
    - Retrait des paramètres d'appel
    - Empilement de la valeur de retour

La Pile Système

- Où se trouve la pile ?
  - A l'adresse indiquée par un registre
- Le noyau est réentrant
  - Le même code peut être exécuté par plusieurs processus utilisateurs en concurrence
- Quand un processus « bascule » en mode noyau
  - Le mot d'état du processeur est remplacé
    - Sauvegarde registres (instruction, pile, ...)
    - Chargement registres pour le mode système **du processus**
    - **Une pile système différente pour chaque processus**
  - **MAIS**
    - **En mode noyau tous les processus partagent le même espace mémoire**
      - **Certaines zones (la plus grande partie) sont vues identiques**
      - **Certaines sont vues différemment (la pile, le contexte propre)**

Architecture des noyaux

- Le noyau étant un programme, il peut être écrit de plusieurs façons
- Noyau monolithique
  - Un unique programme fournit tous les services (gestion des processus, des fichiers…)
  - Tous s'exécute en *kernel space*
  - Exemples : Dos, windows 9x, MacOS < 9, Linux
- Micro noyau
  - Le noyau ne fournit qu'un nombre minimal de services (gestion des processus, de la mémoire et des IPCs)
  - Les autres sont fournis par des programmes utilisateurs (systèmes de fichiers, réseau…)
  - Exemples : AmigaOS, Minix, GNU Hurd, Mach
- Hybride
  - Séparation entre éléments mais tout en kernel space
  - Windows NT/XP

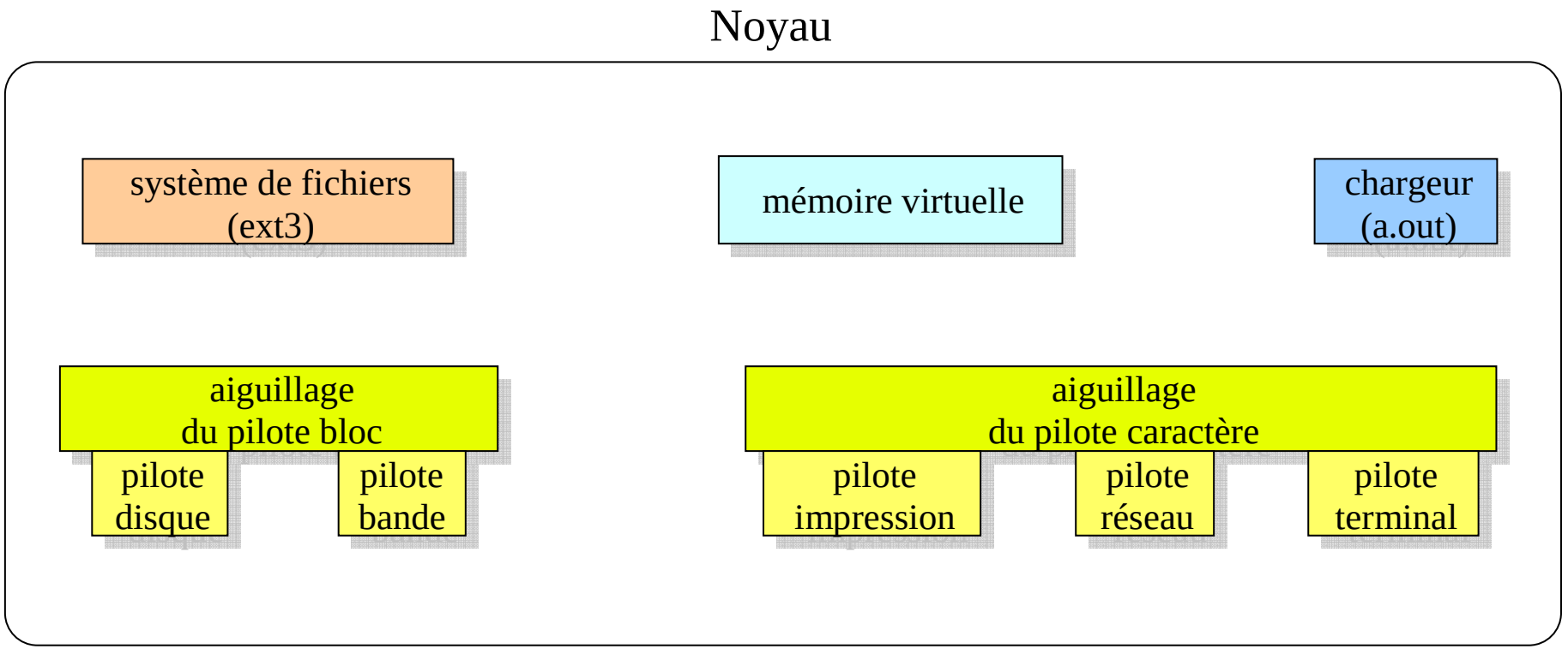
Architecture des noyaux

- Monolithique vs Micro noyaux
  - Une des plus célèbre flamewar d'Internet
  - Andy Tanenbaum vs Linus Torvald
- Andy Tanenbaum
  - Célèbre chercheur et professeur de la Vrije Universiteit, Amsterdam
  - Spécialiste des OS
  - Auteur de Minix, un Unix minimaliste utilisé pour l'enseignement
- Linus Torvald
  - Auteur du premier noyau Linux
  - Étudiant en Master à l'époque
  - Frustré par le manque de développement autour de Minix

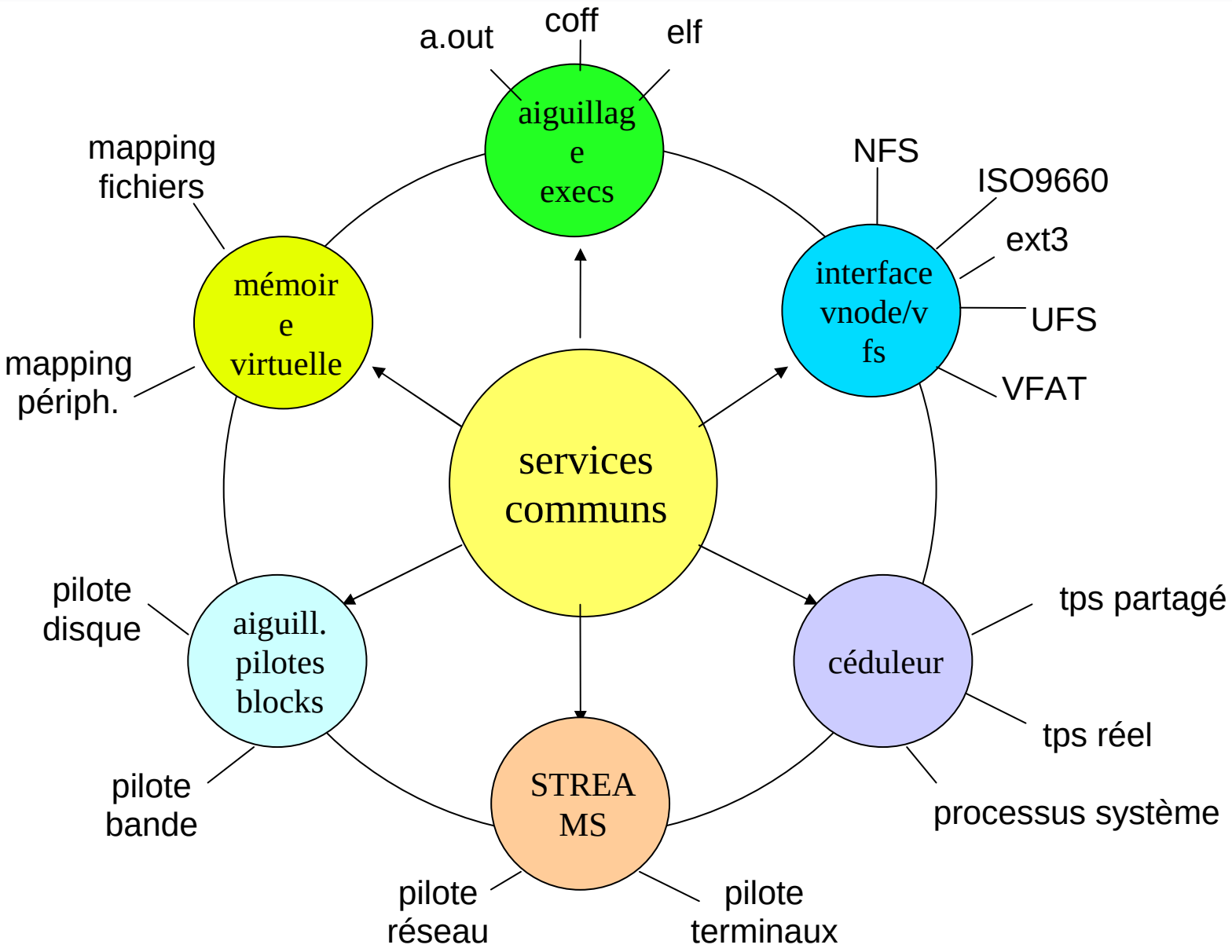
Arguments

- Les micro noyaux sont
  - Plus difficiles à écrire
  - Plus résistants aux bugs (donc plus sures)
- Les noyaux monolithiques sont
  - Plus faciles à écrire
  - Moins élégants que les micro noyaux
  - Plus performants (ou pas)
- Tanenbaum: « This is a giant step back into the 1970s. That is like taking an existing, working C program and rewriting it in BASIC. To me, writing monolithic systems in 1991 is a truly poor idea »
- Torvald : « linux still beats the pants of minix in almost all areas. »

Noyau Unix Traditionnel (en voie de disparition)



Noyau Unix Moderne



Fichier Spéciaux de Périphérique

- Périphériques identifiés par couple <major,minor>
  - Major : famille de périphérique
    - Indique le driver à utiliser
  - Minor : sélection du périphérique dans la famille
  - Exemples :
    - Disque dur IDE : major=3
      - Minor = numéro de partition (0=disque entier)
    - Disque dur SCSI : major=8
- <major,minor> non exploitables directement
  - Utilisation au travers de fichiers spéciaux
    - man mknod
  - Les fichiers spéciaux sont placés dans /dev
    - Par commodité

3. Eléments d'administration

Partitions et montages

- Fichiers spéciaux pour les disques durs
  - /dev/hdaXX, /dev/hdbXX, ... : disques IDE
  - /dev/sdaXX, /dev/sdbXX, ... : disques SCSI
  - /dev/fdaXX, /dev/fdbXX, ... : disquettes
  - ...
- Montage manuel
  - mount -t fstype /dev/xxxx /point/de/montage
    - fstype : ext3, reiser, nfs, vfat, iso9660, ...
- Montage automatique
  - Fichier /etc/fstab
- 6 colonnes
  - 1 : fichier device
  - 2 : point de montage
  - 3 : type de fs
  - 4 : options de montage
  - 5 : dump (sauvegarde)
  - 6 : ordre pour le fsck

Installation de Nouveaux Logiciels

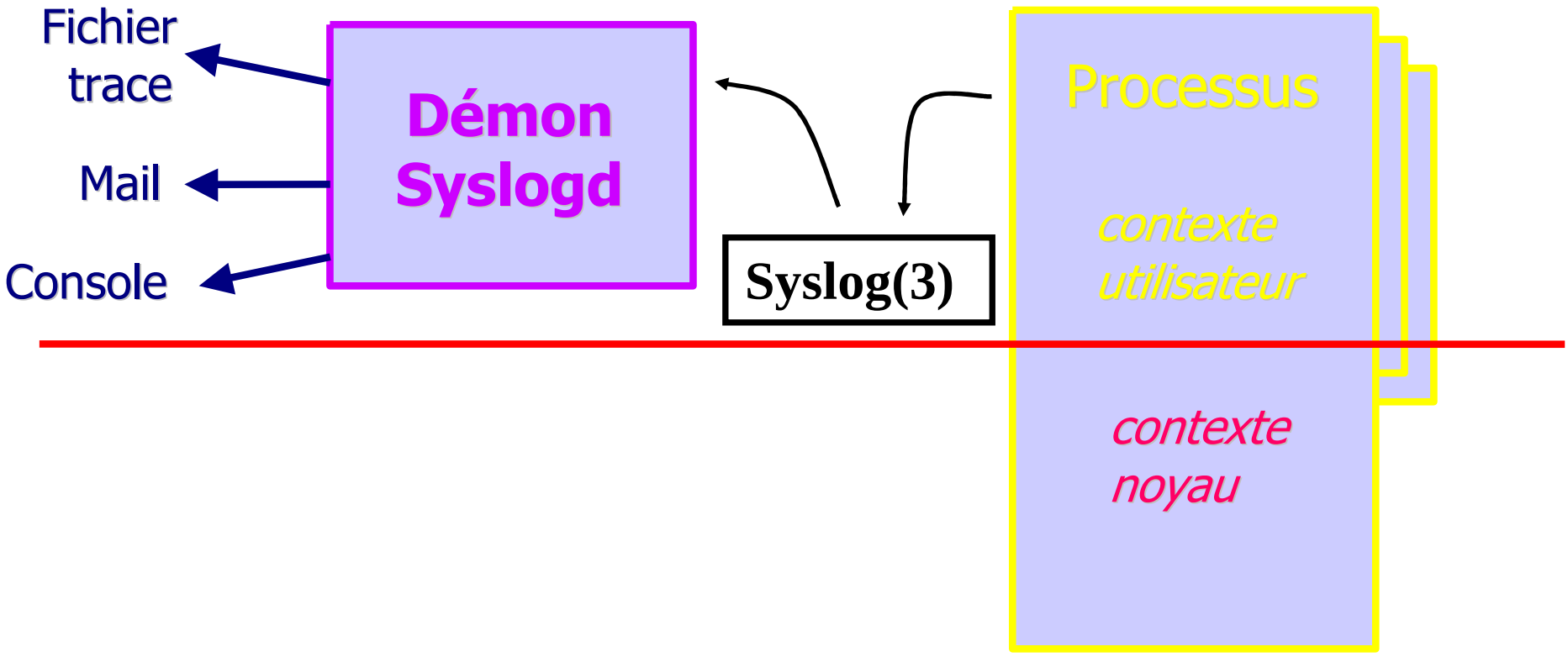
- Méthode « for the braves »
  - Récupérer archive .tar.gz
  - Avec de la chance, configuré avec automake
    - ./configure ; make ; make install
- Paquetages de distribution
  - Principaux formats :
    - .deb (Debian, Ubuntu, ...)
    - .rpm (RedHat, Mandriva, ...)
  - Applications d'installation intelligente
    - .deb : dselect
    - .rpm : rpm

Dépots de Paquetages

- Malgré les dépendances, des conflits peuvent exister
- Informations de dépendances souvent insuffisantes
  - RPM : se contente de réclamer sans préciser où chercher
- Solution : dépôt de paquetages
  - Site web regroupant de nombreux paquetages
    - Cohérents entre eux
    - Mises à jour régulières
  - Commandes de mise à jour
    - Apt (Debian), yum/up2date (RH)
    - **Synaptic** : fonctionne avec les 2 formats
      - Lancement : **`synaptic`**

Observation du système, traces

- Syslogd, le démon bavard



Configuration de Syslog

- Fichier /etc/syslog.conf
  - Lignes formées de couples Sélecteur/Action
    - Sélecteur simple : facility.priority
      - Attention : priority signifie niveau >= priority
      - Ex : mail.notice messages concernant le mail, pour tous les niveaux de notice à emergency (donc sauf info et debug)
    - Sélecteur étendus
      - facility1,facility2.priority
      - facility.=priority; facility.!=priority …
    - Action :
      - /chemin/vers/un/fichier/de/trace, @machine.domaine, user1,user2, * : wall(1) (tous ceux qui sont connectés) …

Exemple de configuration Syslog

- *Tous les messages critiques, sauf ceux du noyau dans un fichier dédié*

```
.=crit;kern.none                /var/adm/critical
```

- *Tous les messages du noyau vers un fichier dédié*

```
kern.*                          /var/adm/kernel
```

- *Tous les messages du noyau sont envoyés vers une autre machine et à la console*

```
kern.crit                       @finlandia
kern.crit                       /dev/console
```

- *Tous les messages du noyau de info a critique dans un fichier dédié*

```
kern.info;kern.!err                     /var/adm/kern.info
```

- *Les messages mail de priorité info vers un tty, les autres vers un fichier*

```
mail.=info                      /dev/tty12
mail.*;mail.!=info                  /var/adm/mail
```

Diagnostiques Réseau

- Commandes utiles :
  - ping : envoie un datagramme ICMP
  - traceroute : recherche une route
  - arp : information sur le cache ARP
  - ifconfig : informations sur interface
  - netstat : information sur l'état du réseau
  - route : information sur le routage
  - tcpdump/snoop : analyser/filtrer les paquets reçus
  - etherfind : tracer les trames
  - rpcinfo : liste des services RPC enregistrés

Chargement dynamique de code noyau

- Les noyaux modernes sont capables de charger dynamiquement du code
  - Code dynamique = « module noyau »
  - Même idée que la liaison dynamique
    - Mais applicable à un programme en cours d'exécution
      - Le noyau bien-sûr
- Les modules peuvent être chargés automatiquement
  - Chargement à la demande
  - Les modules doivent être installés
    - /lib/modules/<version noyau>/...
    - Calcul des dépendances
      - Commande **depmod -a**

Configuration des modules

- Pourquoi configurer ?
  - Les modules peuvent accepter des paramètres
    - No d'interruption, DMA, séléction du matériel, ...
  - Plusieurs modules peuvent fournir un service
    - Cartes son, carte réseau, etc
- Où configurer ?
  - Généralement dans fichier **`/etc/modprobe.conf`**
    - Dépend des distributions

Commandes pour manipuler les modules

- insmod module.o
  - Chargement dans le noyau
- rmmod module
  - Déchargement
- moprobe module.o
  - Chargement d'un module installé
- modinfo module.o
  - Information sur un module
- Depmod
  - (re)Calcule les dépendances entre modules (installés)

Chargeurs de noyau

- Pourquoi donc ?
  - Config multi-boot
  - Charger un noyau est **très compliqué** ...
- LILO (LInux LOader)
  - Le chargeur "historique" (obsolete)
  - Outil Ad Hoc Linux
  - Soit Linux, soit "Other"
  - Quelques limitations gênantes (information dans le MBR...)
- GRUB (GRand Unified Bootloader)
  - Le chargeur générique de GNU/FSF (pour Hurd)
  - Support natif pour multiples OS (Linux, *BSD, ...)

Configuration de GRUB

- Pourquoi (re)configurer GRUB ?
  - Pour installer un nouveau noyau
    - Sans casser une configuration existante qui marche
- Où ça se passe ?
  - Répertoire /boot/
    - Fichier de config /boot/grub/grub.conf
      - section commune
      - entrée 1
      - entrée 2
      - entrée chaînée (autres systèmes)
      - ...

Fichier /boot/grub/grub.conf

- Identifiant des disques et partitions
  - disque : hdn avec n = 0, 1, 2, ...
    - numérotation unique pour IDE et SCSI
      - dans l'ordre de détection (l'ordre dépend de la configuration du/des BIOS)
  - partition : (hdn,x) avec x = 0, 1, 2, ...
- Exemple de fichier de config

```
# section commune
timeout 10
default 0
# première entrée
title Noyau Linux 2.4.17
root (hd1,1)
kernel /boot/vmlinuz-2.4.17 root=/dev/hdb2 read-only
initrd /boot/initrd.img-2.4.17
# deuxième entrée...
```

4. Machines virtuelles

VMware

- Compagnie leader en outils de virtualisation (Virtual Machine) pour x86
- Permet d'exécuter une machine virtuelle x86 sur une architecture x86
- Possibilité d'installer n'importe quel OS dedans
  - Fichiers représentant un disque avec un OS installé
- Plusieurs VMs peuvent s'exécuter en parallèle
- *VMware Player*
  - Gratuit
  - Ne permet pas la création d'images, juste l'exécution
  - Nombreux outils gratuits pour la création
  - Très tatillon sur la configuration ☹
- Utilité
  - Tests/Debugging
  - Migration de programmes
  - Isolation
- La virtualisation est l'avenir!
  - Ajout de support dans le hardware

Principe de VMware

- Fournit un environnement virtuel
  - Carte réseau
  - Carte vidéo
  - Processeur….
- Le hardware est toujours le même
  - Portabilité des programmes
- Machine hôte
  - Machine exécutant le processus VMware
- Machine virtuelle invitée
  - Machine s'exécutant à l'intérieur d'un processus VMware
- Communications
  - Utilisation de TCP/IP pour communiquer
  - La machine invitée utilise la machine hôte comme passerelle
  - Réseau mis en place automatiquement par VMware

Techniques de virtualisation

- Émulation
  - Permet l'exécution d'un programme écrit pour une plateforme x sur une plateforme y
  - Une instruction X est exécutée par une routine qui simule le CPU
  - Lent mais universel
  - Exemple: émulateurs DS, Neo-Geo…
- Recompilation Dynamique (*Virtual PC*)
  - Compile les instructions lors de leur première exécution
- *Virtual Processing (VMware)*
  - Si possible, le code est directement exécuté sur le CPU
  - Sinon, ré-écriture dynamique
  - Permet une exécution à environ 80% de la vitesse

QEMU

- Émulateur de processeur open source
- 2 modes de fonçtionnement
  - *Full system* : Émule un système complet (CPU, périphériques…)
  - *User mode emulation* : Permet de lancer un programme linux compilé pour un CPU sur un autre.
- Émule complètement le CPU…
- En option : *QEMU accelerator module*
  - Fonctionne comme VMware
  - Mais nécessite les droits administrateurs