UNS – Master 1 Informatique

Approfondissement Système

Olivier Dalle

1. Introduction

Organisation du Cours

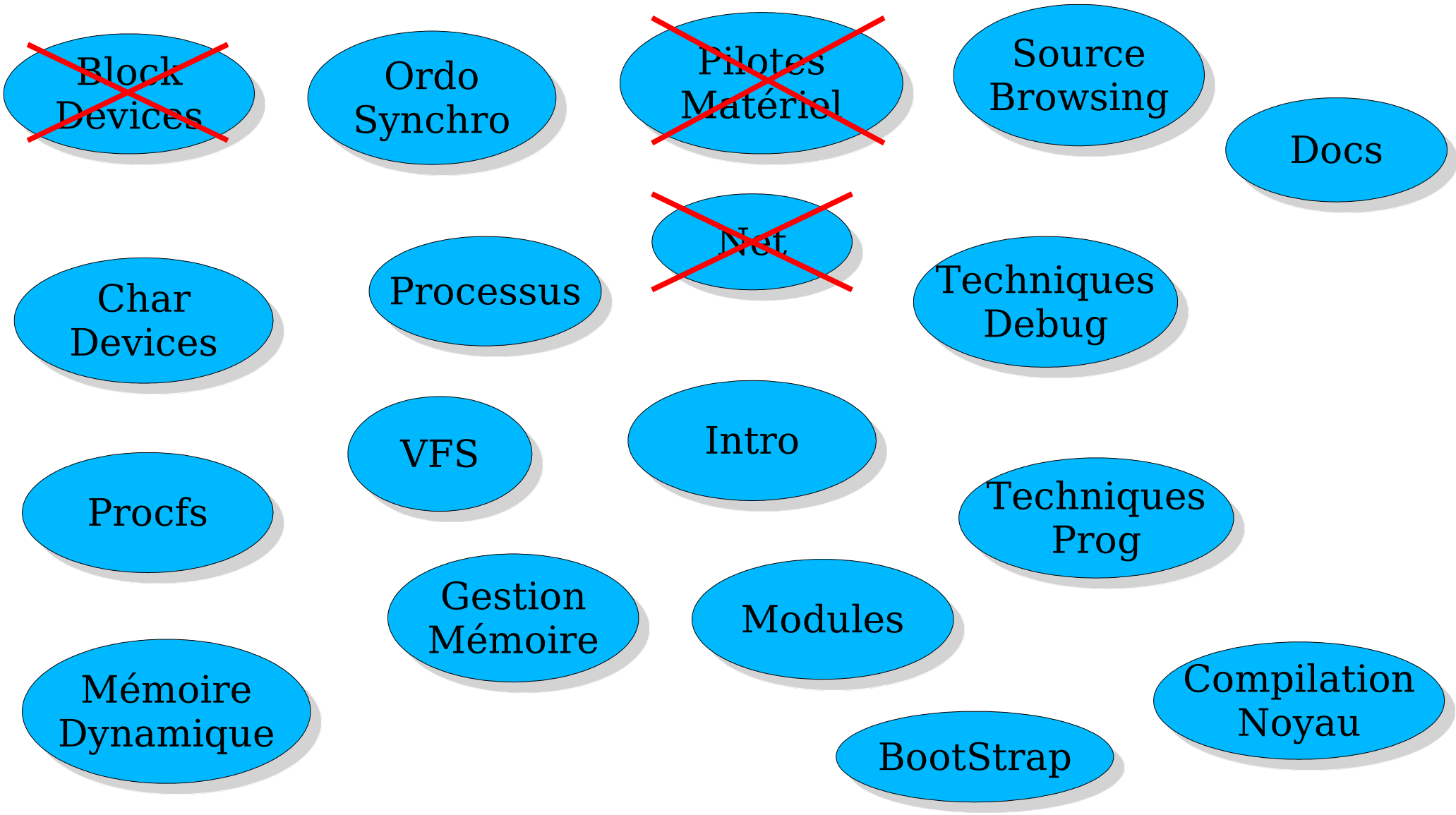
- 12 x 2h Cours
- 12 x 2h TP
  - Utilisation de vmware + Linux Debian 3.1
- Contrôle des Connaissances :
  - Examen : 70%
  - Tps à rendre

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, ...
- Autres ouvrages intéressants
  - **Linux Core Kernel Commentary** (Maxwell, Coriolis)
  - **Linux Kernel Internals** (Beck, Addison-Wesley)
  - + Classiques : Bach (SysV), McKusick (BSD 4.4), ...

Autres Ressources

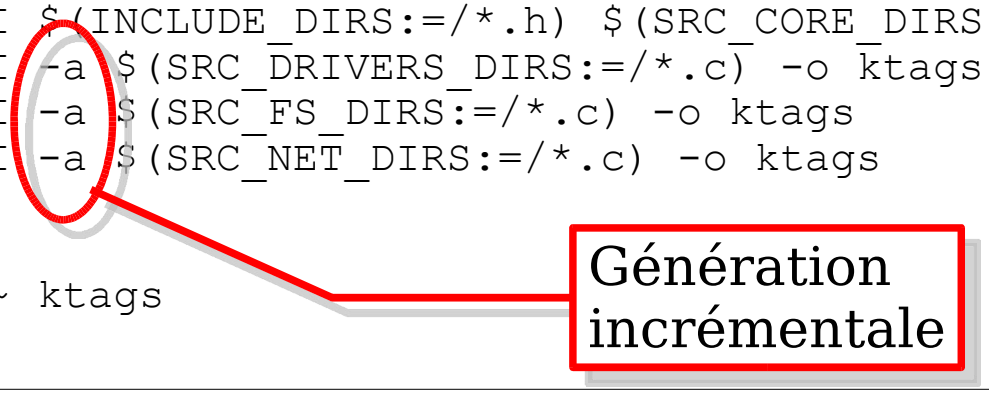
- Ressources du Web
  - http://www.linuxhq.com/lkprogram.html
    Point de départ très utile, nombreux liens
  - www.google.com ...
- La documentation du noyau
  - /usr/src/linux/Documentation
  - Génération DocBook des sources du noyau
    - Cibles Makefile principal :
      - make psdocs pdfdocs htmldocs
- 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 tq 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 vmware
  - Emulation fiable et robuste d'un PC et de son BIOS
  - Nombreux avantages
    - Snapshot/clones, partage ressources avec hôte, ...
- Pour les TP
  - distrib Debian 3.1
  - Noyau Linux 2.4.X
  - Login `root`
    - pwd : `root;;`
  - Sources linux installés et recompilés dans /usr/src
    - Au démarrage choisir le noyau « tpasy »

2. Rappels Système

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 ...

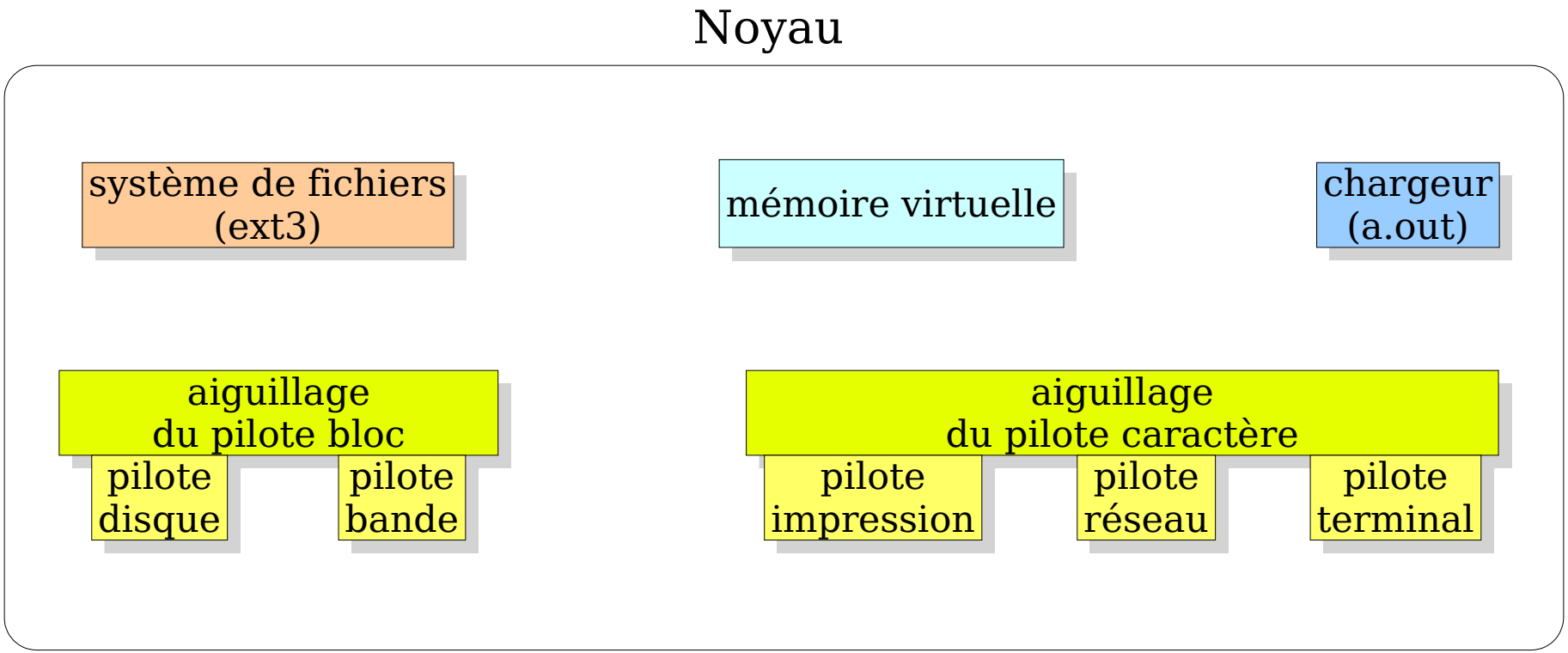
Qu'est-ce que le noyau ?

- Le noyau est un **programme** spécial
  - S'exécute directement « sur » le matériel
  - 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)

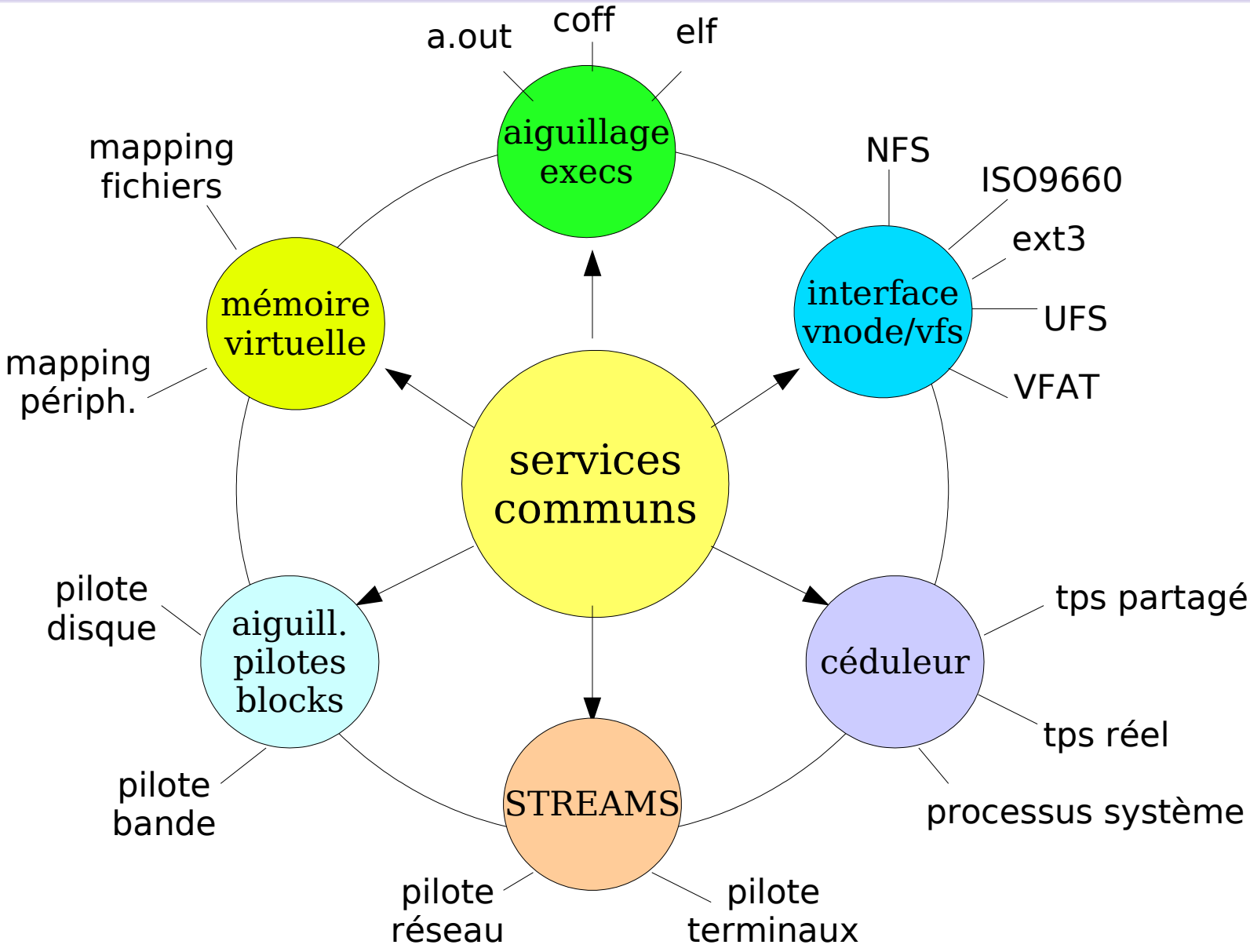
Que fait le noyau ?

- 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)

Noyau Unix Traditionnel (en voie de disparition)



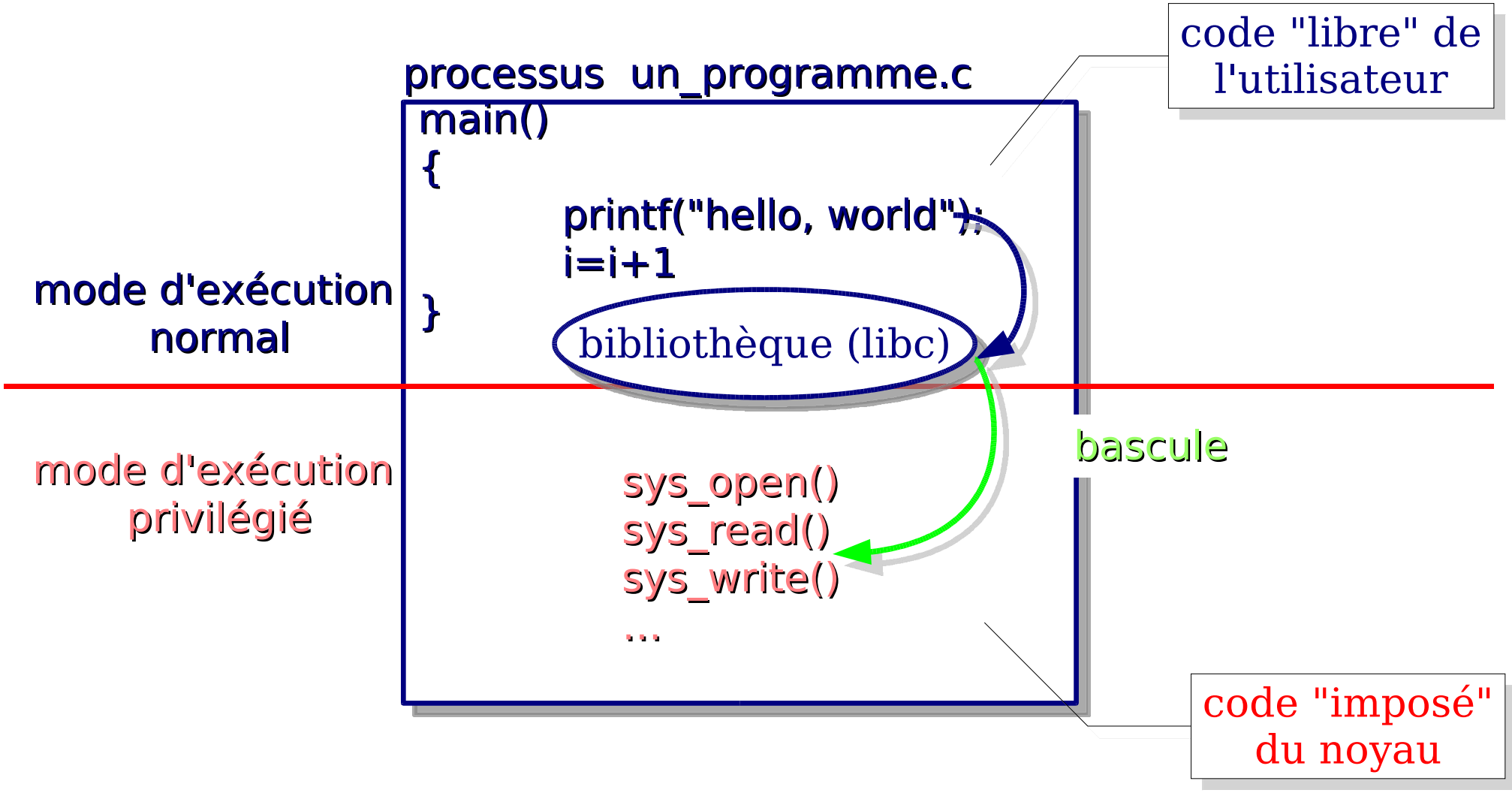
Noyau Unix Moderne



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

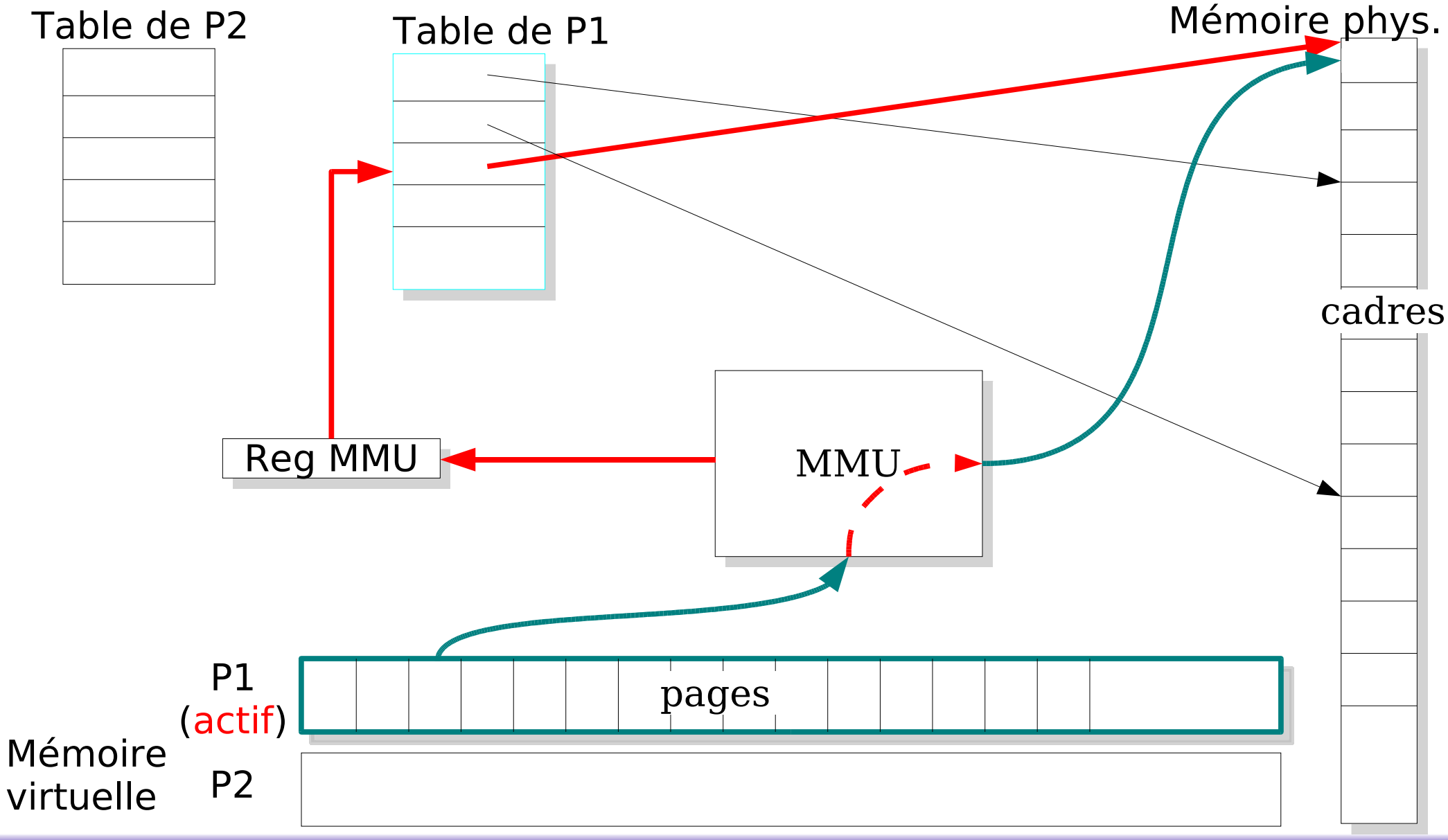
Illustration : Passage en Mode Noyau



Mémoire Virtuelle

- 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

Principe de Fonctionnement du MMU



La Pile

- 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
- 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)

Configuration Réseau (TCP/IP)

- Pour converser avec une autre machine, il faut connaître :
  - Sa propre adresse de liaison
  - Sa propre adresse IP
  - L'adresse IP de la machine cible
  - L'adresse de liaison de la machine cible

Adresses de l'émetteur

- Adresse IP initialisée par :
  - Système : ifconfig (Unix)
  - PROM : variable PROM à affecter (Terminaux X)
  - Réseau : obtention de son adresse par demande auprès d'un serveur RARP ou DHCP (Diskless, TX)
- Adresse Liaison
  - Fournie par le constructeur :
    - Adresse unique (en principe) délivrée dans une PROM
  - Elle peut être (re)configurée
    - Plage d'adresses réservée aux reconfigurations locales
    - Certaines cartes peuvent être reconfigurées avec toute adresse, y compris celles réservées aux constructeurs

Adresse du destinataire

- Adresse IP
  - Fournie par l'application qui établit la connexion :
    - L'utilisateur donne directement l'adresse IP
    - L'utilisateur donne un nom logique machine.domaine.TLD
      - Un répertoire d'adresses (fichier /etc/hosts) permet la conversion locale
      - Un service réseau d'annuaire (NIS, DNS) fournit le service de conversion
- Adresse de Liaison
  - Obtenue à l'aide du protocole de résolution ARP
    - L'émetteur diffuse unpaquet d'interrogation
      - Soit le destinaire le reçoit et répond
      - Soit un routeur se fait passer pour le destinataire

Adresses de Diffusion

- Chaque réseau (sous-réseau) possède sa propre adresse générique de diffusion
  - permet de joindre toutes les machines du réseau/sous-réseau en même temps
  - Adresse = tous les bits de la partie machine à 1
  - Variante (pas conventionnelle) : tous les bits de la partie machine à 0
    - Tous les bits à 0 sont utilisés par un expéditeur pour indiquer qu'il ne connaît pas sa propre adresse (RARP, DHCP)
- Une adresse de diffusion globale : 255.255.255.255

Fichier Spéciaux de Périphérique

- Périphériques identifiés par couple <major,minor>
  - Major : famille de péripérique
  - 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

Installation de Nouveaux Logiciels

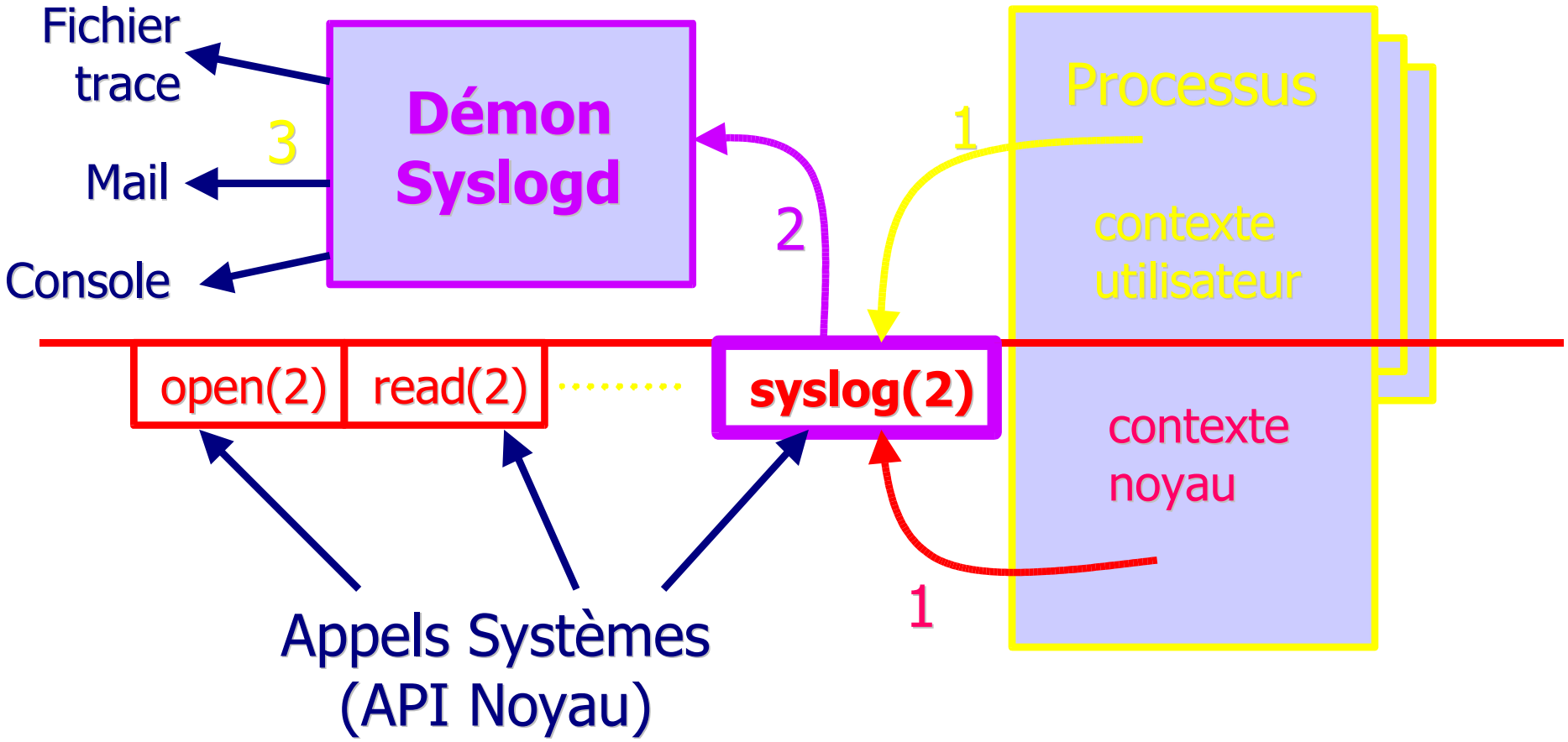
- Méthode « for the braves »
  - Récupérer archive .tar.gz
  - Avec de la chance, configuré avec automake
    - ./configure ; make ; make install
- Paquetages de distribution
  - Principaux formats :
    - .deb (Debian, Ubuntu, ...)
    - .rpm (RedHat, Mandriva, ...)
  - Applications d'installation intelligente
    - .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 ?
  - Ca dépend de la distribution ...
  - Généralement dans fichier **`/etc/modprobe.conf`**
  - **Sur Debian :**
    - Fichiers dans répertoire /etc/modprobe.d/

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
- GRUB (GRand Unified Bootloader)
  - Le chargeur générique de GNU/FSF (pour Hurd)
  - Support natif pour multiples OS (Linux, *BSD, …)
  - Nombreuses améliorations

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...
```