Administration de Systèmes UNIX

Gaël DELBARY

Reprise des cours de Frédéric COMBEAU

Qu'est-ce que l'Administration de Systèmes ?

- **Les différentes actions d'un administrateur système**
  - Gérer les comptes utilisateurs
  - Gérer les impressions
  - S'occuper des sauvegardes et des restaurations
  - Tuner et surveiller les systèmes
  - Assurer la sécurité
  - Mettre à jour le système (patch ?, update ?, upgrade ?)
  - Installer les produits
  - Gérer l'espace disques
  - Arrêter et redémarrer le système
  - Surveiller le réseau (et le réparer ou l'améliorer)
  - Installer de nouveaux systèmes et de nouveaux matériels
  - Réparer les problèmes qui surviennent tout seul
  - Écrire des scripts pour automatiser un maximum de choses
  - Assister à des réunions
  - Répondre aux questions diverses des utilisateurs
  - ...

Définitions

Quelques Rappels de Bases

- **Notion de LOGIN**
  - Identification / Authentification
- **Le HOME ou le répertoire utilisateur**
- **Les variables d'environnement**
  - PATH, PWD, USER, HOME, LANG, UID, ...
- **Les commandes de base et le SHELL**
  - `cd, pwd, ls, cp, mv, rm, mkdir, id, man ...`
  - `sh, csh, ksh, bash, zsh, ...`
- **Les briques de bases**
  - La combinaison de commandes (bash)
    - le « || », le « && », le « & »
  - La re-direction
    - les « < », « > », « 1> », « 2> », « 2>&1 », « | »

Remenber me

Quelques Rappels de Bases

- **Notion de processus**
  - Exécutable en cours d'exécution
  - Traité par le SCHEDULER *(mode Kernel vs mode User)*
- **Les segments d'un exécutable**
  - CODE : contient le code exécutable
  - DATA : données globales initialisées
  - BSS : données globales non-initialisées
  - HEAP (ou tas) : données dynamiques
  - STACK (ou pile) : données locales, fonctions, paramètres, ...
- **Exécutables statiques versus exécutables dynamiques**
  - Notions de librairies, de symboles et de chargement dynamique
- **Les threads**
- **Cycle de vie d'un processus**
  - Le fameux FORK/EXEC
  - Les différents états : R (runable), S (sleeping), D (non interruptible), T (stracé), Z (zombie), W (si swappé), N (si nicé)

Remenber me

Quelques Rappels de Bases

- **La gestion de la mémoire et les goulets d'étranglement**
  - Mémoire virtuelle versus mémoire physique
  - Espace d'adressage
  - Mémoire partagée
  - Mémoire cache
  - Buffers

- **La pagination**
  - Plus petite unité de mémoire gérée par le gestionnaire de mémoire virtuelle : la page
  - Les processus réclament des pages de mémoires
  - Peuvent-ils en libérer ?

- **Le swap**
  - Quand il y a pénurie de pages mémoires, un processus entier est swappé (transféré sur disque dur) pour libérer des pages mémoires

Remenber me

PLAN (enfin ...)

- **Historique des UNIX (pour y voir un peu plus clair)**
- **Concepts de base sur l'administration Unix**
  - L'arborescence UNIX
    (ou comment retrouver son chemin ?)
  - Les partitions
    (mount, devices, RAID et LVM, les systèmes de fichiers)
  - Les différents types de fichiers
    (ou comment découvrir le pays des fichiers)
  - Boot loader et procédure de boot matériel
    (ou de la mise sous tension jusqu'à l'exécution du noyau)
  - Démarrage et arrêt d'un système UNIX
    (ou de l'exécution du noyau jusqu'au prompt login)
  - Les démons et le lancement de services
    (ou comment lancer des chevaux de Troie)
  - Les processus, la mémoire virtuelle et les entrées/sorties
    (ou comment tuner un système)

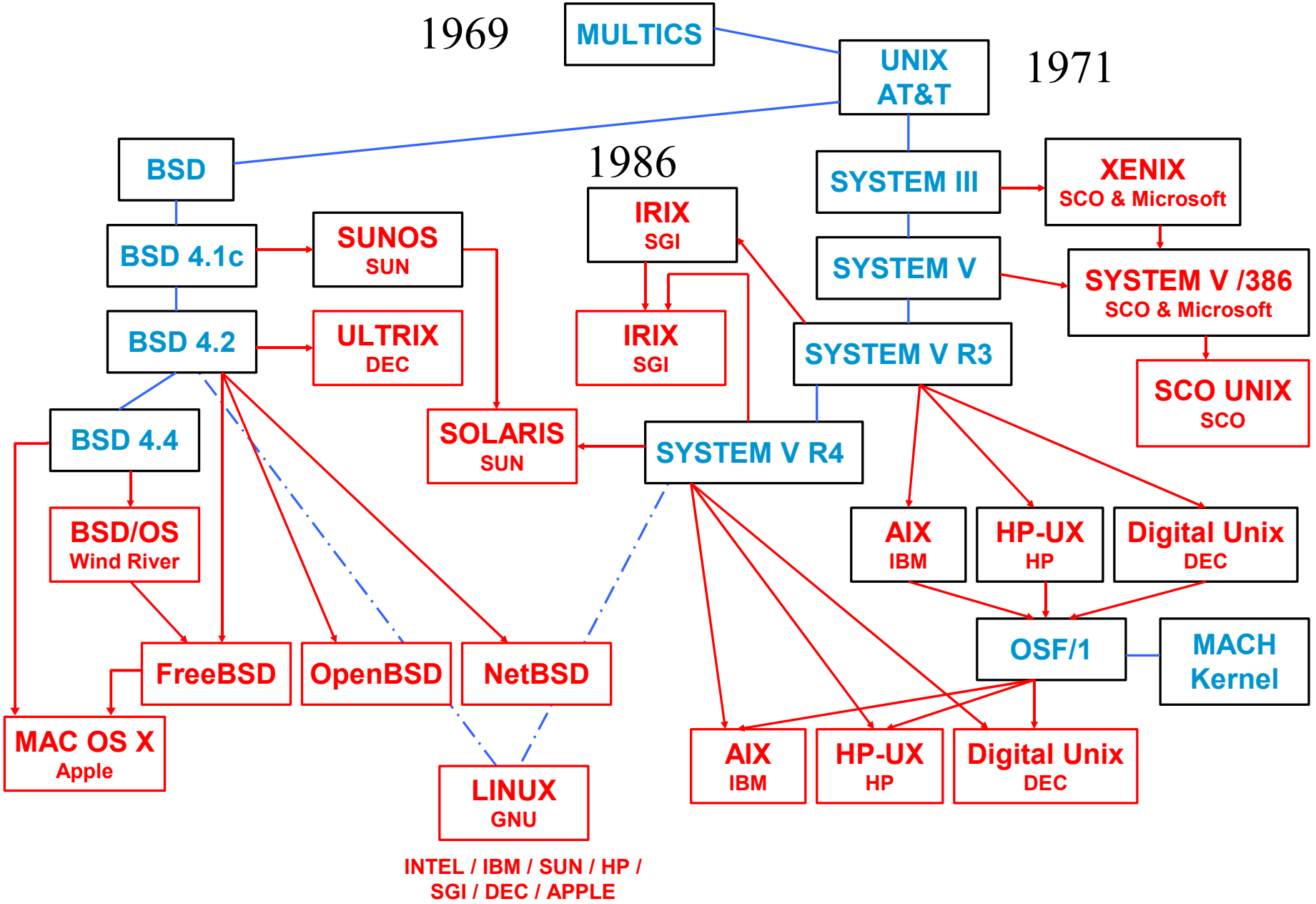
PLAN (enfin ...)

- **Exploitation d'un système Unix**
  - Les comptes utilisateurs (ou comment autoriser l'utilisation du système)
  - L'authentification (password, groups, shadow et PAM)
  - Les permissions sur les fichiers, les quotas disque et les ACLs (ou comment contrôler qui accède à quoi)
  - Exécution décalée : cron, at et les scripts d'exploitation (ou comment automatiser son travail)
  - Le noyau : fonctionnement, modules, configuration et compilation (ou comment empêcher sa station de booter)
  - X-Window (ou comment profiter d'une interface graphique)
  - Sauvegardes et restaurations (ou comment prendre une assurance pour son système)
  - Les impressions (ou comment s'amuser de longues heures avec les imprimantes)
  - Le réseau (ou comment configurer les services de base pour communiquer avec le monde entier)
  - Syslog (ou comment jouer à Big Brother)

Historique des UNIX

Comment y voir un peu plus clair ?

Historique des UNIX



Il était une fois ...

Concepts de Base de l'Administration Système

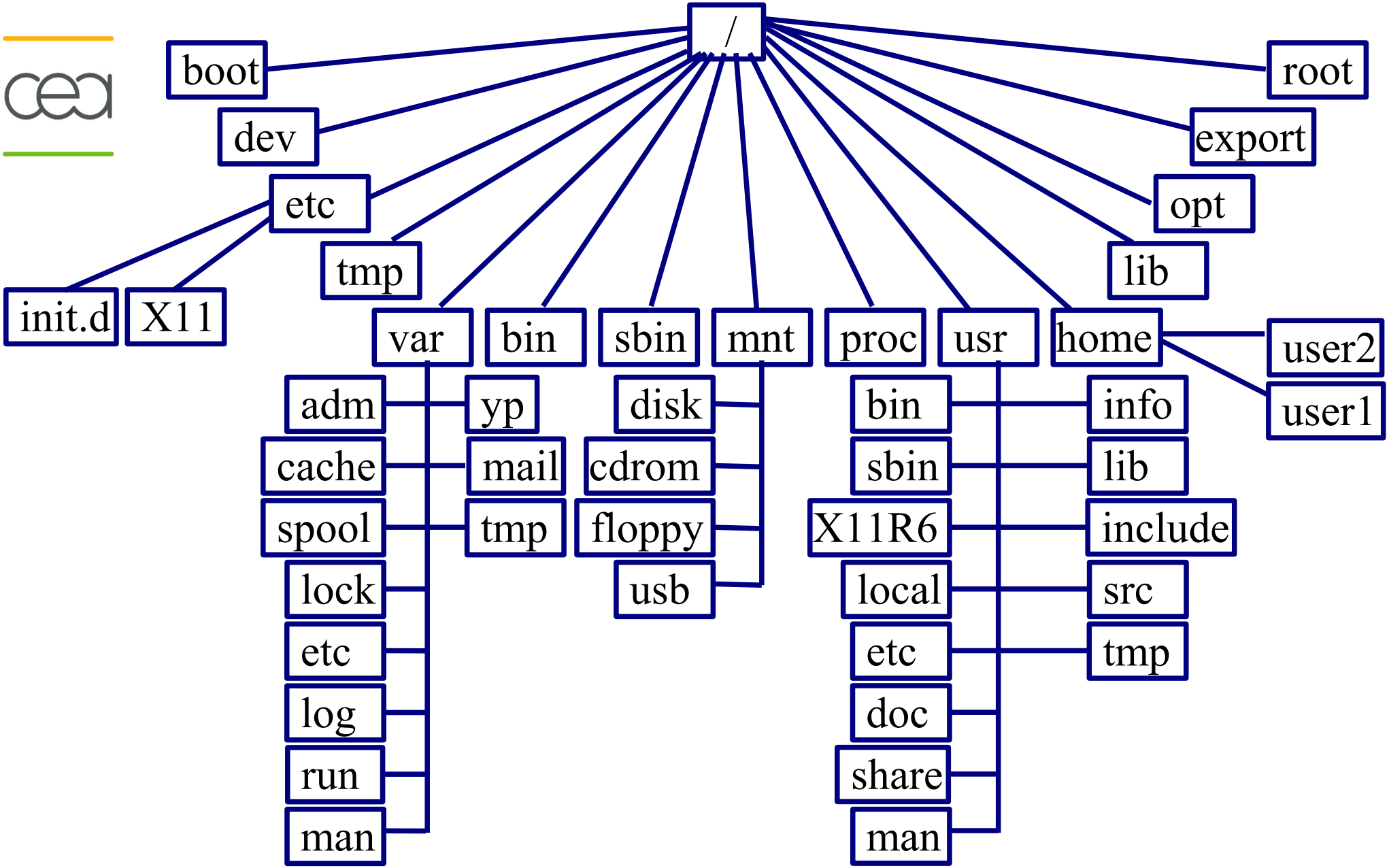
Arborescence UNIX

Arborescence Générale

- **Tout système Unix contient une et une seule arborescence**
- **Ne pas oublier que dans un système Unix : TOUT EST FICHIER**
- **Cette arborescence contient tout ce dont a besoin un système UNIX pour fonctionner**
  - Un noyau et ses modules
  - Des fichiers pour accéder aux devices
  - Des fichiers de configuration
  - Des binaires exécutables
  - Des librairies
  - Des répertoires pour accueillir des fichiers temporaires
  - Des fichiers de log
  - Des répertoires pour gérer les impressions
  - Des répertoires pour les utilisateurs
  - Des produits tiers (binaires, configurations, librairies, ...)

TOUT EST FICHIER.

Arborescence Générale



Branches et arbres

Arborescence Générale

- /boot : configuration de boot + noyau
- /dev : répertoire de périphériques
- /etc : répertoire des fichiers de configuration
- /tmp : répertoire de fichiers temporaires
- /var : répertoire de fichiers variables (log, impression, mail)
- /bin : exécutables communs
- /sbin : exécutables d'administration
- /mnt : répertoire de montage de périphériques amovibles
- /proc : répertoire d'accès aux données noyau
- /usr : applications globales au système
- /home : répertoires des comptes utilisateurs
- /lib : répertoire des librairies de base
- /opt : répertoire d'installation des produits tiers
- /export : répertoire des partages réseau
- /root : répertoire de l'administrateur

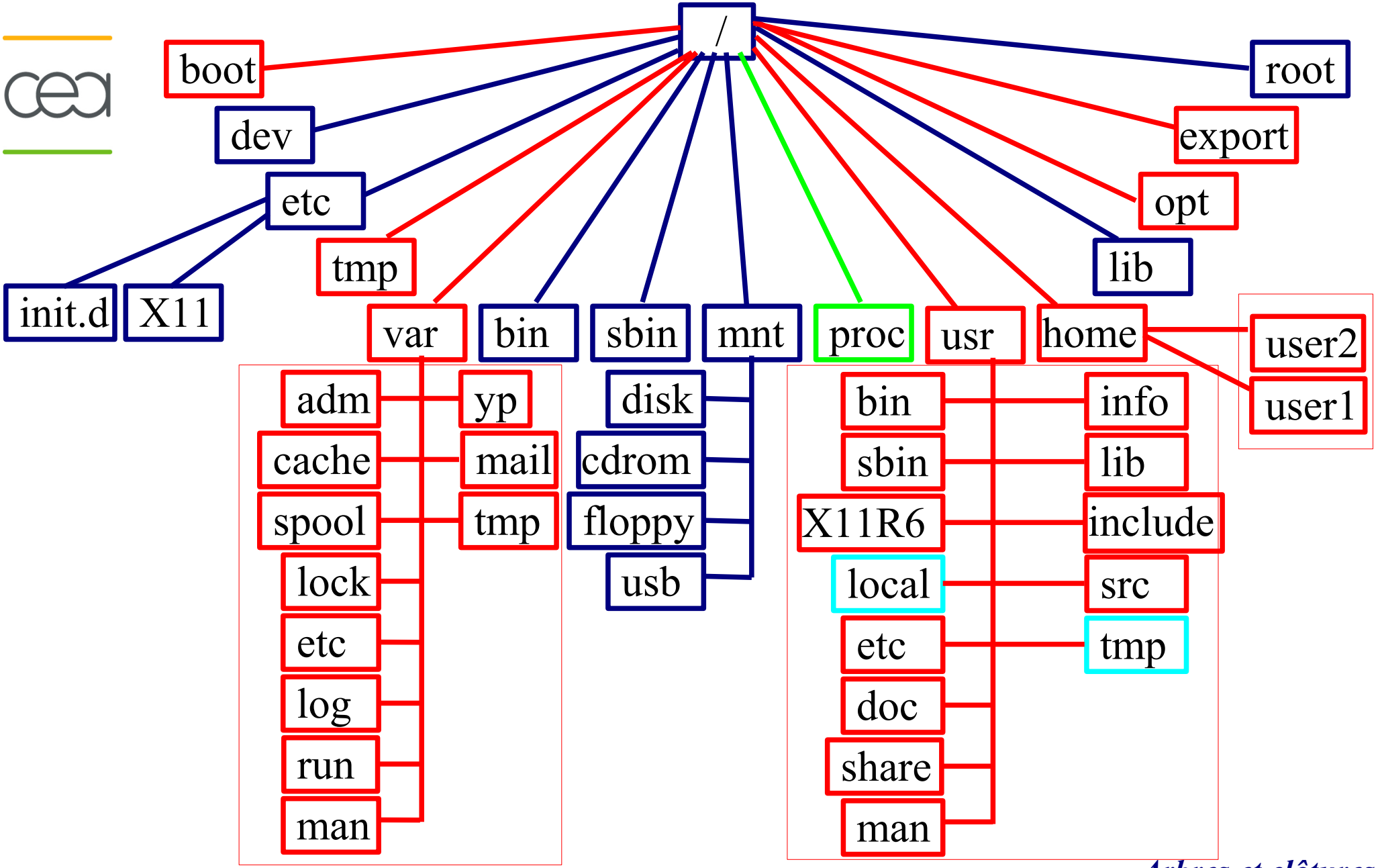
- lost+found : i-node non recouvrable lors d'un fsck

A quoi ça sert ?

Concepts de Base de l'Administration Système

Partitions, devices, RAID et LVM, systèmes de fichiers

Arborescence et Partitions



Arbres et clôtures

Arborescence et Partitions

- **Les répertoires pouvant être mis sur des partitions**
  - /boot : limitation du bios
  - /tmp : les fichiers temporaires
  - /var : les fichiers qui changent beaucoup
  - /usr : tous les utilitaires utilisateurs
  - /usr/tmp : fichiers temporaires utilisateurs
  - /usr/local : produits tiers
  - /home : répertoires utilisateurs
  - /opt : produits tiers
  - /export : répertoires partagés en réseau
  - /proc : variables noyau + processus

  - *TOUT le reste peut aller dans la partition de /*

  - *Sans oublier la partition de swap !*

- **Mais c'est quoi une partition ?**

Arbres et clôtures

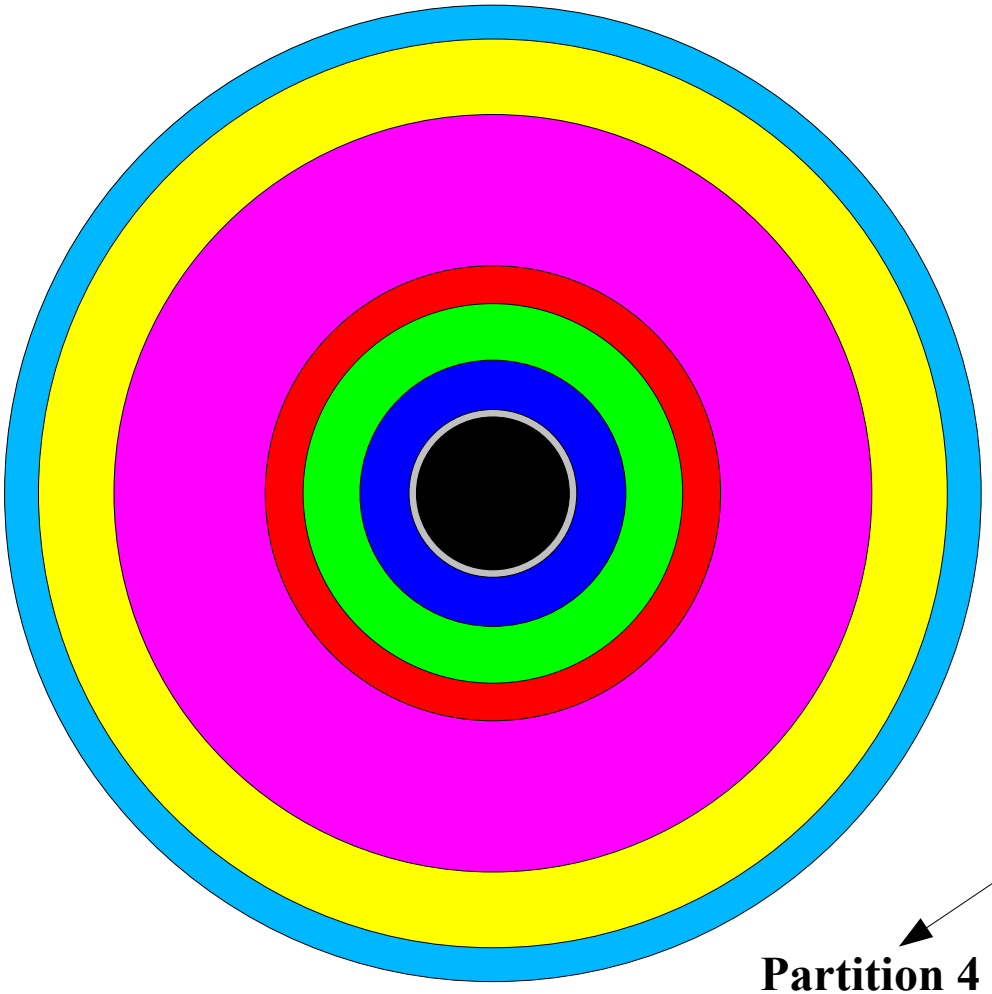
Partitions

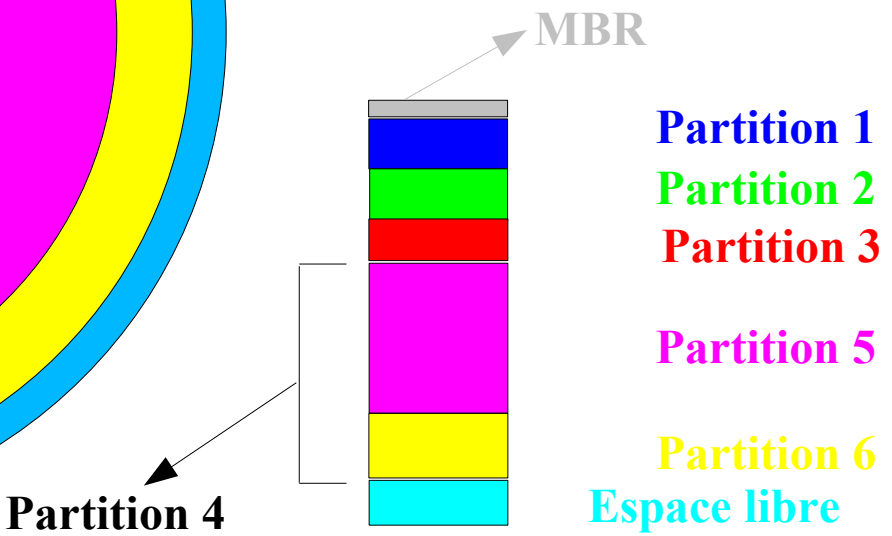
- **C'est une partie d'un périphérique de stockage (disque dur, ...)**
- **Tout périphérique de stockage est découpé en partition**
- **Il existe deux types de partitionnement**
  - Le partitionnement DOS
    - Contient un MBR (Master Boot Record)
    - 4 partitions primaires
    - Toutes les partitions sont séquentielles
    - Notion de partition étendue et partition logique
  - Le partitionnement BSD
    - 8 ou 16 partitions selon les Unix
    - Les partitions peuvent se recouvrir
    - La partition C définit la totalité du périphérique

Découpons les disques

Le partitionnement DOS

Partitions primaires
    Partitions 1,2 et 3
Partition étendue
    Partition 4
Partitions logiques
    Partitions 5 et 6

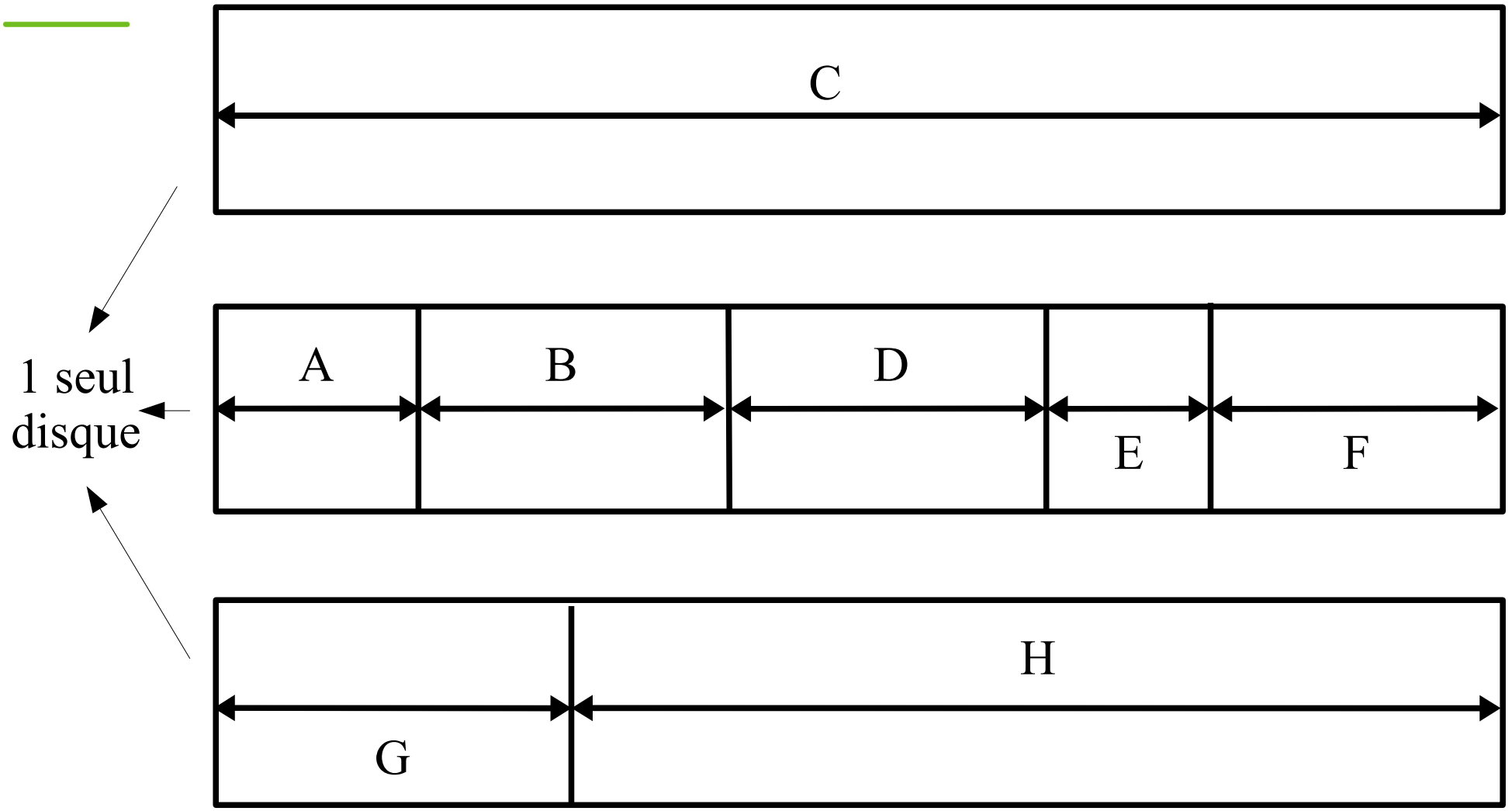




PC et Dos

- **Chaque partition DOS**
  - Est soit une partition primaire, une partition étendue ou une partition logique
  - A un début et une fin, exprimés en nombre entier de cylindres
  - A un identifiant, permettant de repérer le type de système de fichiers qu'il contiendra (mais peut être différent)
  - A un boot flag qui est positionné ou pas (défini la partition bootable par défaut)

Le partitionnement BSD



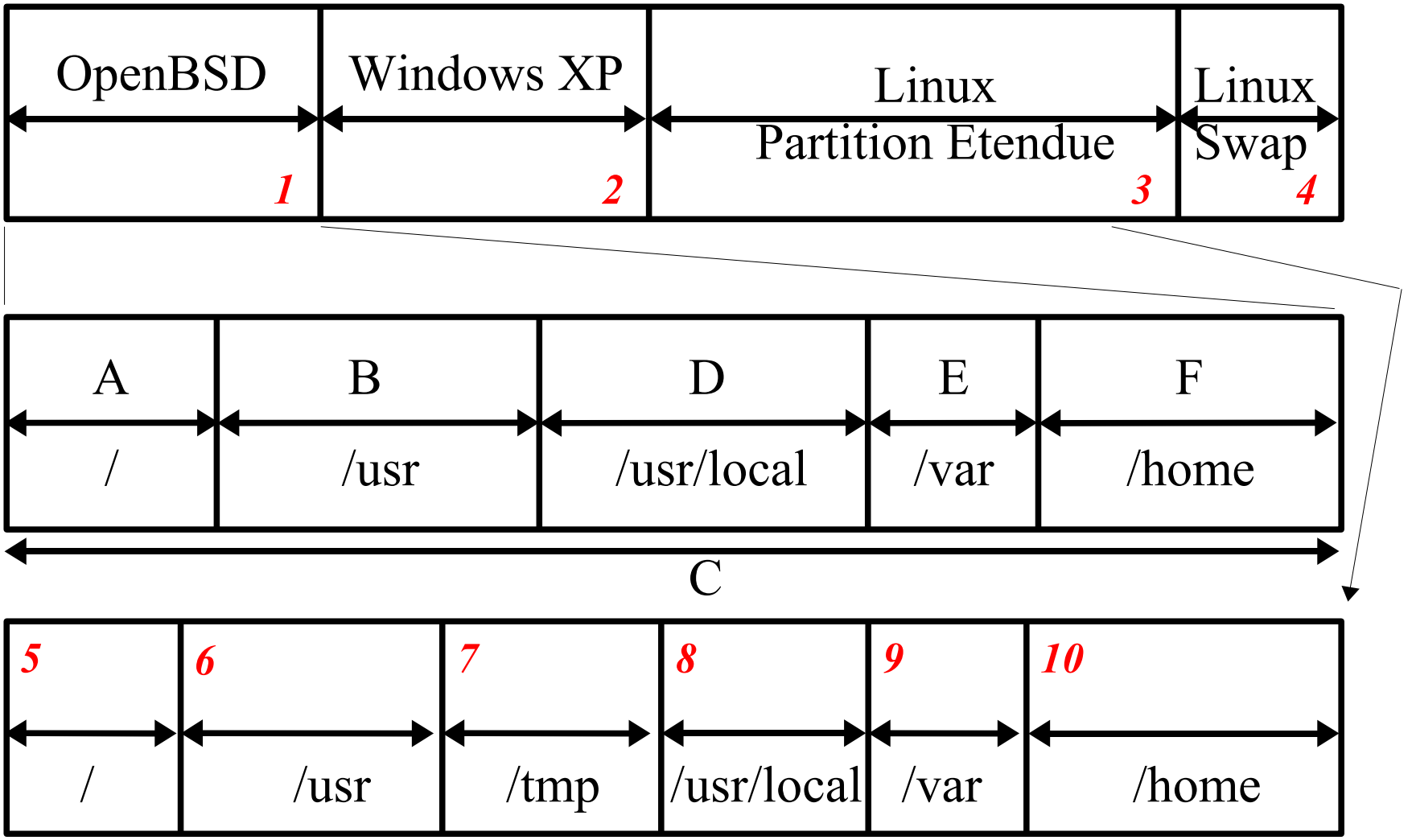
Partitions BSD

Partitions

- **Chaque partition BSD**
  - A un début et une fin, exprimés en nombre entier de cylindres
  - Peut avoir un type, permettant de repérer le type de système de fichiers qu'il contiendra (mais peut être différent)
  - Peut avoir un label, pour lui donner un nom
- **Les disques BSD sont partitionnés au formatage**
- **Ils sont, donc, déjà partitionnés lorsqu'on les reçoit, mais on peut les re-partitionner**
- **Pas de notion de MBR**
- **Mais le nombre de partitions est limité par le système**

- **Et les systèmes BSD sous PC, comme FreeBSD, quel type de partition utilisent-ils ?**
  - Ils peuvent utiliser un système de partition à la BSD
    - Mais le disque sera dédié au système d'exploitation BSD
  - Mais si on veut pouvoir mixer plusieurs systèmes d'exploitation
    - Par exemple : Windows XP, Linux et OpenBSD
    - On utilisera un système de partitionnement mixte
      - Le disque dur est partitionné au format DOS
      - Mais une partition DOS sera partitionné au format BSD

- **Le partitionnement BSD sur un PC avec plusieurs OS**



Partitions

- **Les partitions sont les contenants pour les systèmes de fichiers**
- **Après le partitionnement, il faudra y créer un système de fichiers**
- **Mais comment accéder à ces partitions ?**
  - Grâce aux fichiers du répertoire /dev
- **Les fichiers présents dans /dev permettent l'accès aux partitions et à l'ensemble du disque**
- **Pour certains Unix, l'accès aux partitions peut s'effectuer de 2 manières**
  - Par les fichiers en mode bloc (standard)
  - Par les fichiers en mode caractères (pour les sauvegardes)

Partitions

- **Le nommage des fichiers d'accès aux partitions dépend de l'Unix utilisé**

- **Pour Linux**
  - Disque IDE : hd [a|b|c|...] [1|2|3|4|...]
    - a : contrôleur primaire, disque master
    - b : contrôleur primaire, disque slave
    - c : contrôleur secondaire, disque master
    - d : contrôleur secondaire, disque slave
    - Chiffre : numéro de partition pour un disque donné
  - Disque SCSI/SATA : sd [a|b|c|...] [1|2|3|4|...]
    - a : premier device vu sur les bus scsi/sata
    - b : deuxième device vu les bus scsi/sata
    - Chiffre : numéro de partition pour un disque donné
  - Exemple :
    - */dev/hda1* : première partition du premier disque (master) du premier contrôleur

Partitions

- **Pour Solaris**
  - Disque en mode bloc : /dev/dsk/c[n]t[n]d[n]s[n]
  - Disque en mode caractère : /dev/rdsk/c[n]t[n]d[n]s[n]
    - c[n] : numéro du contrôleur (ide ou scsi)
    - t[n] : numéro du device sur le bus scsi ou ide
    - d[n] : numéro de sous-device scsi (presque toujours 0)
    - s[n] : numéro de partition correspondant à un partitionnement BSD
      - 1 : partition a
      - 2 : partition b
      - 3 : partition c ...
  - Exemples :
    - *c0t0d0s3* : tout le disque d'identifiant scsi 0 sur le contrôleur 0
    - *c1t0d0s1* : partition a pour le disque primaire du premier contrôleur ide
    - *c2t1d0s3* : partition c pour le disque secondaire du second contrôleur ide (lecteur CD-ROM)

- **Pour OpenBSD**
  - Disque en mode bloc
    - /dev/wd[n][m] pour IDE
    - /dev/sd[n][m] pour SCSI
    - Où n représente le numéro du disque dur avec une partition DOS OpenBSD valide (slice OpenBSD)
    - Et m la lettre de la partition BSD dans ce slice OpenBSD
  - Disque en mode caractère
    - /dev/rwd[n][m]
    - /dev/rsd[n][m]
  - Exemples :
    - */dev/wd0c* : premier disque IDE (master) du premier contrôleur
    - */dev/wd0a* : première partition du premier disque (master) IDE du premier contrôleur
    - */dev/wd2b* : deuxième partition (souvent le swap) du premier disque (master) IDE du second contrôleur
    - */dev/cd0c* : premier lecteur de CD-ROM vu sur la chaîne IDE

RAID

- **Une partition n'est pas forcément une partie d'un disque dur, ce peut être beaucoup plus compliqué que cela**
- **Principalement à cause du RAID et des LVMs**

- **Le RAID (Redundant Array of Independent Disks)**
  - Permet de combiner des disques en un seul (ou plusieurs) vu(s) par le système d'exploitation
  - Les disques logiques vus par l'OS peuvent avoir une gestion interne spécifique
  - Cette gestion des disques physiques réalisée par l'interface peut être de différents niveaux, définis par des nombres
  - 9 niveaux : RAID-0 à RAID-7
    - Seuls les niveaux 0, 1 et 5 sont réellement utilisés
  - Deux types de RAID peuvent être utilisés :
    - RAID matériels
    - RAID logiciels
  - Buts du RAID : Assurer l'intégrité et la disponibilité des données

Esquisser le RAID

RAID

- **RAID matériel se compose**
  - D'une carte d'entrée/sortie qui comporte un bus (ide ou scsi)
  - Sur lequel on installe des disques durs
  - Stockés dans une baie externe

- **C'est la carte d'entrée/sortie qui réalisera les écritures/lectures physiques sur les disques durs**
- **Le système d'exploitation ne voit que les périphériques logiques définies par la carte d'entrée/sortie**
- **Par exemple, le système d'exploitation ne verra que le device *`/dev/sdb`* (sur lequel, on pourra faire des partitions), alors que 3 disques composent la batterie RAID**
- **Lors d'une reconstruction ou d'un problème sur les disques, c'est la carte d'entrée/sortie qui ferra tout le travail**
- **Rapide et ergonomique mais cher**

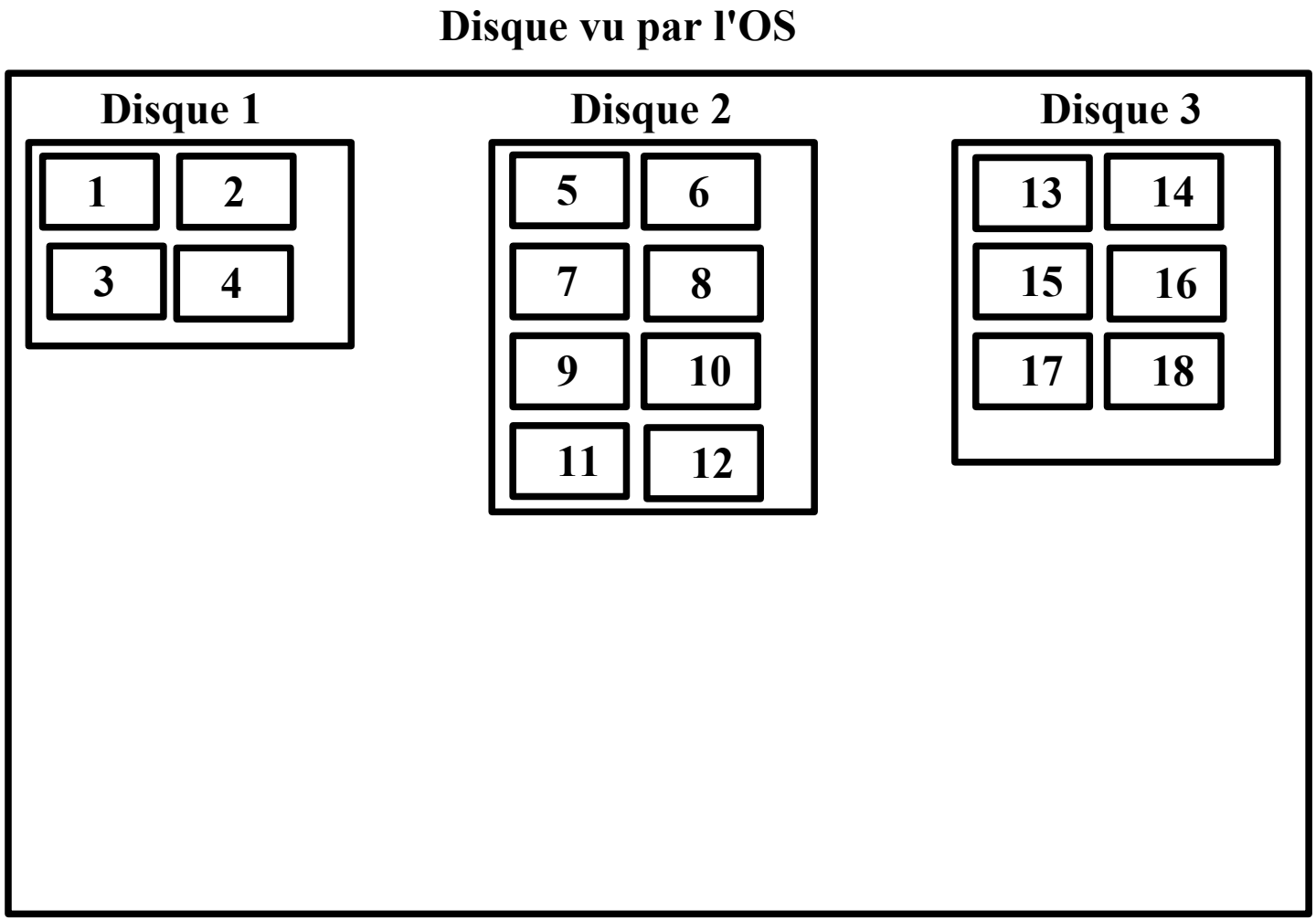
RAID

- **RAID logiciel se compose**
  - De la couche logiciel dans le noyau de l'OS
  - De disque durs connectés au système (SCSI ou IDE)
  - On peut trouver des baies SCSI, vu par l'OS, comme autant de disques durs qu'elles en contiennent

- **C'est la couche logiciel qui gère tout et qui définira un nouveau device pour accéder au regroupement des disques durs**
- **Tout se configure via le système d'exploitation (et plus par la carte matériel)**
- **Comme les couches RAID sont gérées par l'OS, pour y accéder, le noyau doit être chargé**
- **Lors d'un problème sur la batterie RAID, il peut être nécessaire d'intervenir manuellement pour gérer les disques RAID**
- **On peut définir un niveau de RAID sur des partitions (et pas seulement sur des disques durs entiers)**
- **Pas cher (car intégré à l'OS) mais pas forcément rapide ni ergonomique**

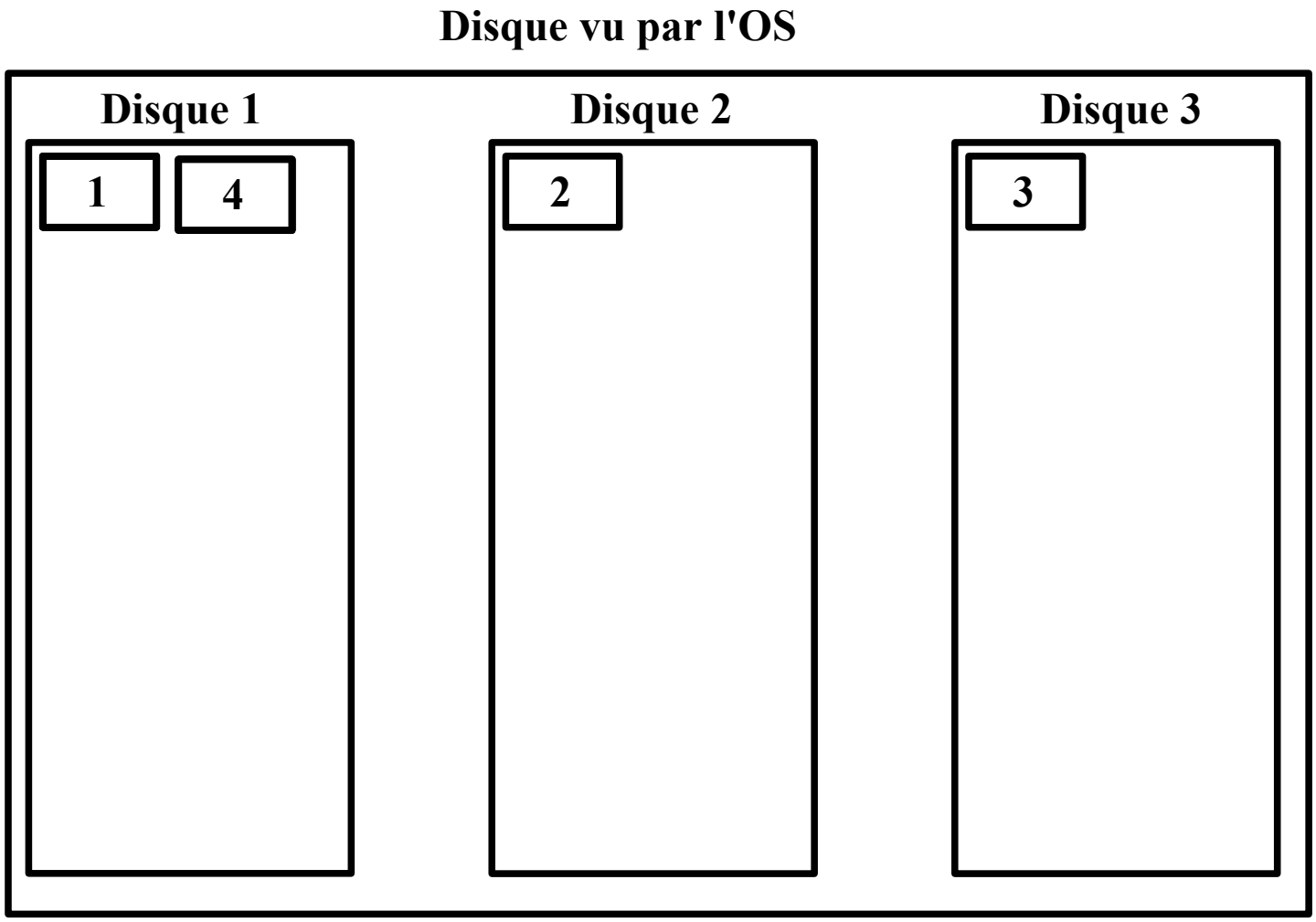
Esquisser le RAID

- **Les différents niveaux de RAID**
  - Linéaire ou JBOD : concaténation de disques (Just a Bunch Of Disks)
    - N disques sont transformés en un seul par concaténation



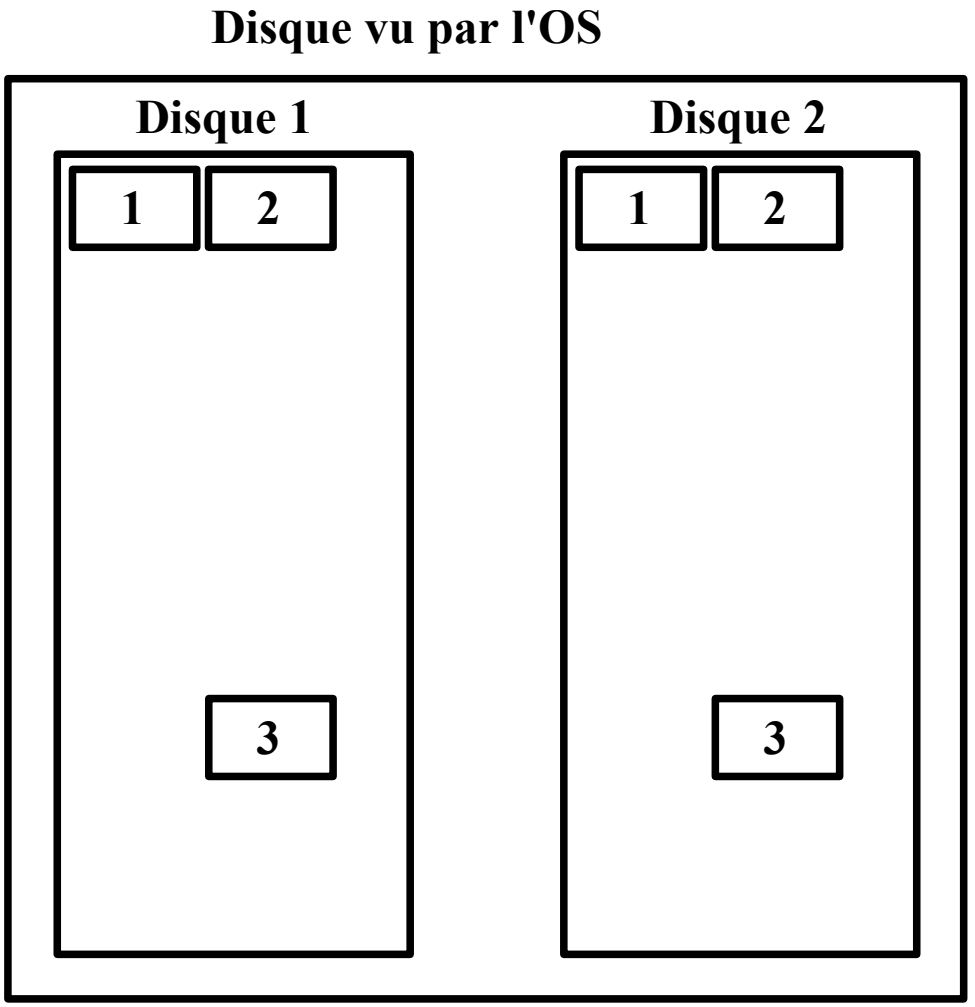
Linéaire

- **Les différents niveaux de RAID**
  - RAID-0 : Fusion de disques (stripping)
    - N disques sont transformés en un seul avec répartition des écritures



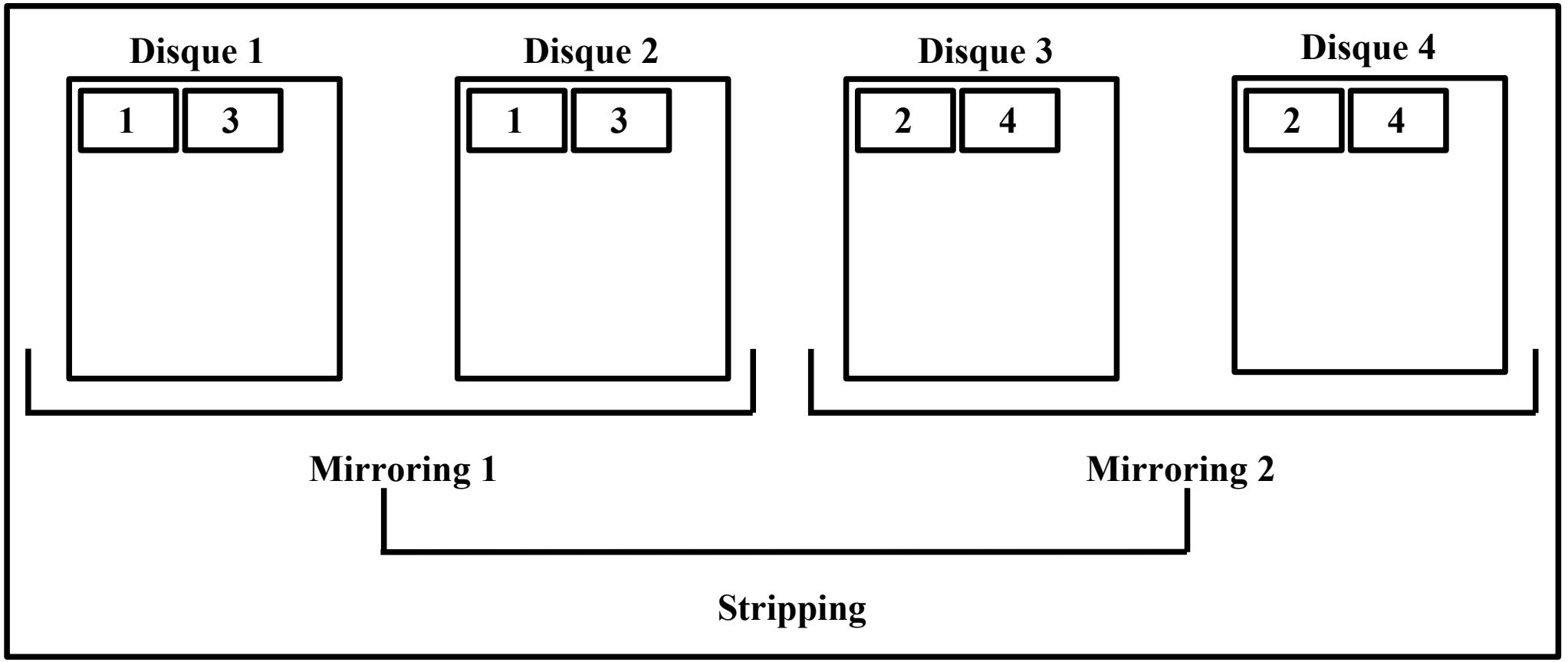
RAID-0

- **Les différents niveaux de RAID**
  - RAID-1 : Mirroring de disques
    - 2 disques sont répliqués secteurs à secteurs



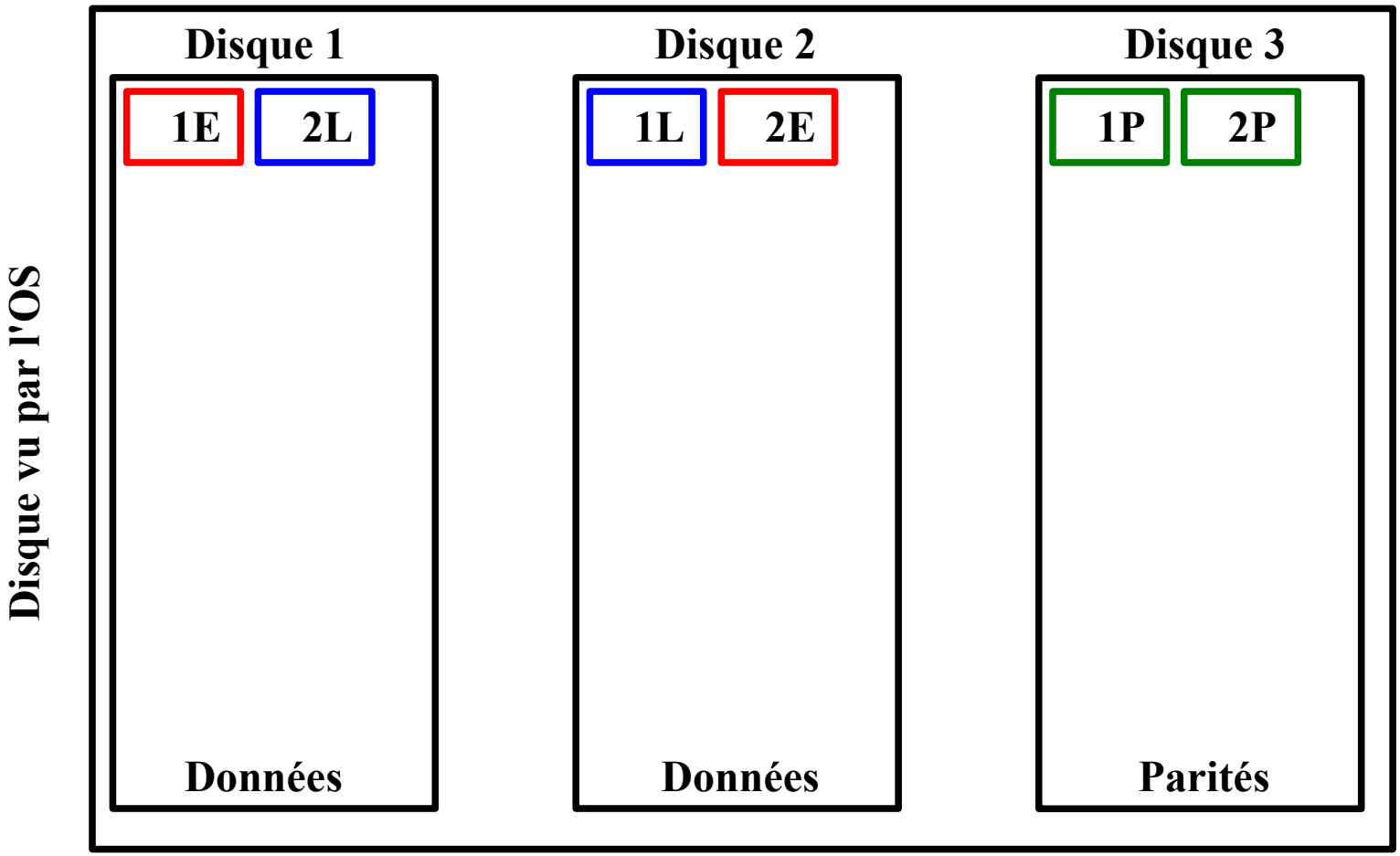
RAID-1

- **Les différents niveaux de RAID**
  - RAID-0+1 : Stripping+Mirroring de disques
    - Un nombre pair de disques sont répliqués 2 à 2 puis fusionnés pair par pair



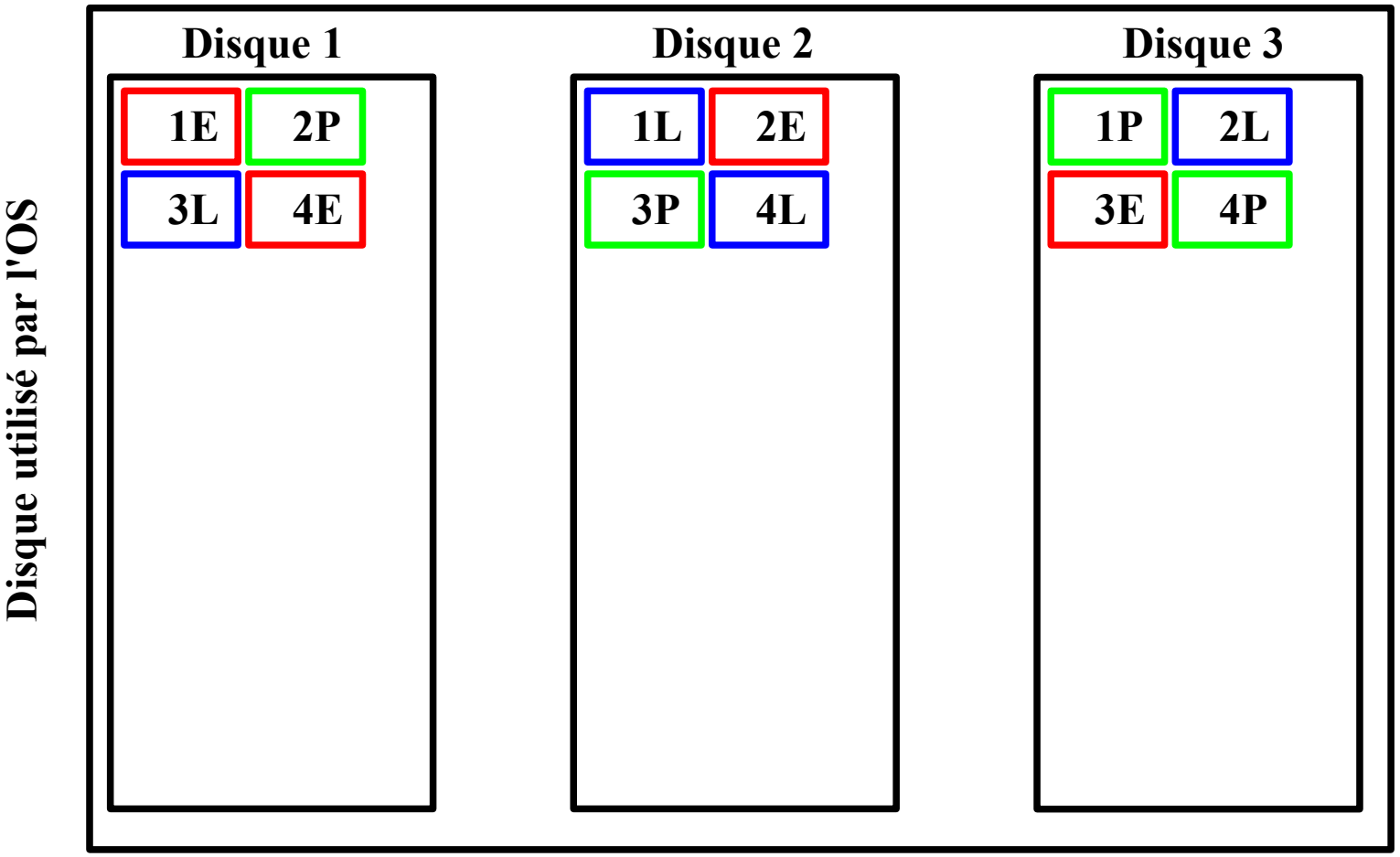
RAID-0+1=?

- **Les différents niveaux de RAID**
  - RAID-4 : Fusion de disques avec disque de parité dédié
    - Sur 3 disques, 2 disques comportent les données avec écriture tournante, le troisième comporte un XOR des 2 premiers disques, bloc à bloc



RAID-4

- **Les différents niveaux de RAID**
  - RAID-5 : Fusion de disques avec données de parité tournante
    - Sur 3 disques, tous les disques comportent des données et les parités sont calculées en tournant



RAID-5

- **Les niveaux de RAID peu utilisés**
  - RAID-2
    - Code de parité sur les bits d'un disque
      - Code de correction d'erreurs de Hamming
      - Les disques durs modernes intégrent la correction d'erreurs
  - RAID-3
    - Identique au RAID-4 mais sur les octets d'un disque (et non les blocs)
    - RAID-4 plus efficace à cause de l'effet cache des blocs
  - RAID-6
    - Deux disques de parités (P et Q) nécessaires
    - 4 disques durs minimum
    - Permet la perte de 2 disques sans indisponibilité
  - RAID-DP
    - Idem RAID-6, disques de parité fixes
    - Propriétaire NetApp
    - Performances égales au RAID-0

- **Exemple d'utilisation du RAID avec Linux**
  - Nom des devices créés
    - */dev/md?*
  - Fichier de configuration
    - */etc/raidtab*
  - Commandes
    - *mkraid* : création d'un array RAID
    - *raidstart* : lancement du device RAID
    - *raidsetfaulty* : défini un composant comme indisponible
    - *raidhotremove* : arrêt d'un composant d'un array
    - *raidhotadd* : ajout d'un composant d'un array
    - *mdadm* : couteau suisse de la gestion des devices RAID
  - Monitoring
    - *dmesg* ou */var/log/messages*
    - */proc/mdstat*
  - Attention au boot avec un système utilisant du RAID logiciel
    - Pourquoi ?

Exemple un peu RAID

Les LVMs (Logical Volume Management)

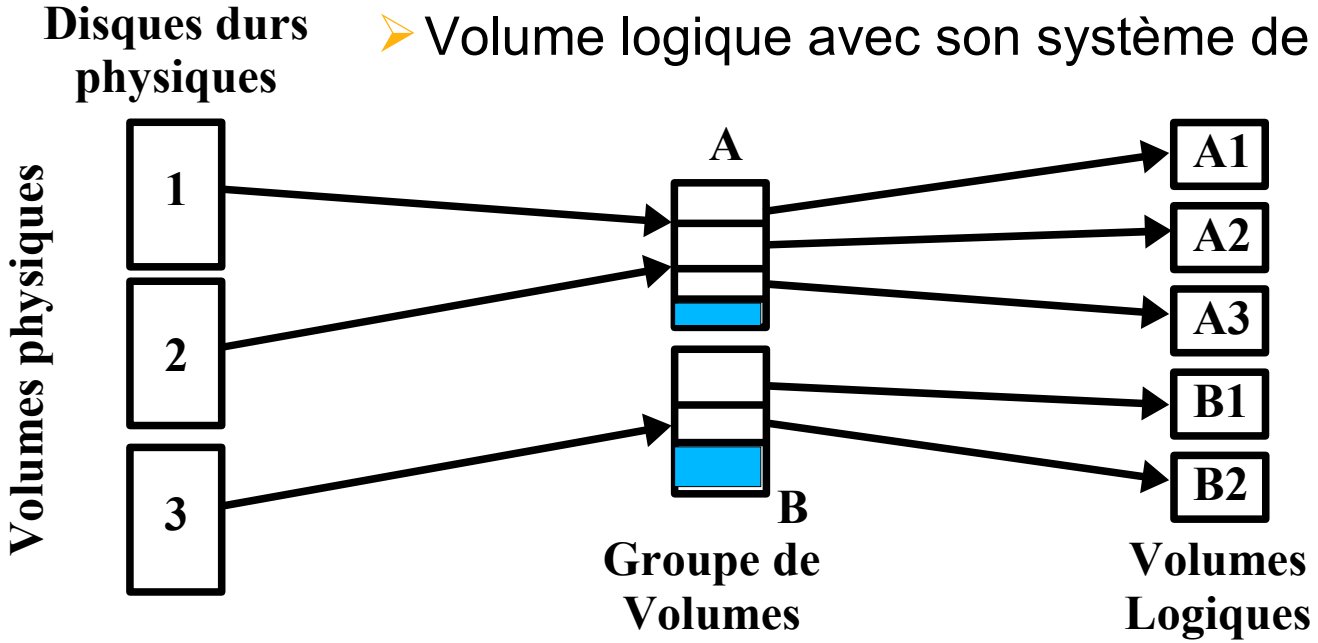
- Concept difficile à appréhender (mieux vaut oublier tout ce que je viens de dire sur les partitions)
- N'existe pas sous tous les OS
- Peut être une option payante dans l'OS
- Les différentes implémentations peuvent être très différentes
- Peut être combiné à du RAID (matériel et/ou logiciel, certaines implémentations du LVM inclus du RAID logiciel)
- Est, essentiellement, logiciel (donc gérer par l'OS)
- Très pratique quand on a compris comme ça marche

- Permet :
  - De s'affranchir des tailles physiques des disques durs
  - D'augmenter dynamiquement la taille d'un système de fichiers
  - D'ajouter ou enlever un disque dur sans devoir tout re-installer
  - De créer des snapshots pour assurer des backups cohérents

LVM = compliqué

- **Les concepts des LVMs**
  - Tout d'abord, on trouve les disques physiques (ou, pour certaines implémentations, les partitions disques, comme on vient de les voir)
  - Un disque physique (ou une partition) doit être initialisé pour définir un volume physique
  - Un volume physique va constituer un groupe de volumes
  - Un groupe de volumes regroupe un ou plusieurs volumes physiques (donc un ou plusieurs disques)
  - Les partitions d'un groupe de volumes se nomment les volumes logiques (les volumes logiques vont abriter les systèmes de fichiers)
  - Un groupe de volumes est constitué d'unités d'espace allouable (partitions physiques ou étendues physiques)
  - Un volume logique est constitué de partitions physiques que l'on peut allouer dynamiquement
  - Un système de fichiers peut, donc, être augmenté à la volée

- **Les concepts des LVMs**
  - Enfin, à la création d'un groupe de volumes, on peut y inclure les algorithmes logiciels de RAID (notion de partitions logiques composés de partitions physiques)
  - En résumé :
    - Disque dur physique
    - Partition de disque dur
    - Volume physique
    - Groupe de volumes
    - Volume logique avec son système de fichiers



LVM = pas si compliqué

- **Exemple d'utilisation de LVM : LVM2 sous Linux**
  - Noms des devices
    - */dev/groupe_de_volume/volume_logique*
  - Fichier de configuration
    - Il n'y en a pas
  - Commandes :
    - *pvcreate*, *pvremove*, *pvscan*, *pvdisplay*, *pvmove* : gestion des volumes physiques
    - *vgcreate*, *vgremove*, *vgscan*, *vgdisplay*, *vgextend*, *vgreduce* : gestion des groupes de volumes
    - *lvcreate*, *lvremove*, *lvscan*, *lvdisplay*, *lvextend*, *lvreduce* : gestion des volumes logiques
    - *resize2fs* : modification de taille d'un système de fichiers
  - Attention au boot de systèmes avec du LVM
    - Pourquoi ?

Partitions et Arborescence

- **Maintenant que les partitions sont créées, où définit-on les points de montage ?**
- **Cela dépend des Unix, mais, en règle général, il s'agit d'un fichier texte sous /etc**
- **Ce fichier définit les fichiers device (dans /dev), le point de montage, les paramètres de montage et les paramètres de sauvegarde (dump) et de vérification d'intégrité (fsck)**

- **Sous Linux**
  - */etc/fstab*
    - */dev/hda6 / ext3 defaults 1 1*
    - Champs 1 : partition de type bloc
    - Champs 2 : point de montage
    - Champs 3 : type de système de fichiers
    - Champs 4 : paramètres de montage de la partition
    - Champs 5 : fréquence de sauvegarde par dump
    - Champs 6 : ordre de vérification d'intégrité au reboot par fsck

fstab

Partitions et Arborescence

- **Sous Solaris**
  - */etc/vfstab*
    - */dev/dsk/c0t3d0s0 /dev/rdsk/c0t3d0s0 / ufs 1 no rw*
    - Champs 1 : partition de type bloc
    - Champs 2 : partition de type caractère (pour les sauvegardes)
    - Champs 3 : point de montage
    - Champs 4 : type de système de fichiers
    - Champs 5 : ordre de vérification d'intégrité par fsck
    - Champs 6 : est-ce que mountall doit monter cette partition ?
    - Champs 7 : paramètres de montage de la partition

- **Sous OpenBSD**
  - */etc/fstab*
    - */dev/wd0a / ffs rw 1 1*
    - Champs 1 : partition de type bloc
    - Champs 2 : point de montage
    - Champs 3 : type de système de fichiers
    - Champs 4 : paramètres de montage de la partition
    - Champs 5 : fréquence de sauvegarde par dump
    - Champs 6 : ordre de vérification d'intégrité (fsck)

Création de systèmes de fichiers

- **Une fois les partitions construites, il faut les remplir**
- **Les remplir avec des systèmes de fichiers**
- **Un système de fichiers permet de gérer**
  - Une arborescence
  - Différents types de fichiers
  - Des droits sur ces fichiers
  - Les propriétaires
  - Les groupes propriétaires
  - L'allocation de nouveaux fichiers ...
- **Il existe plusieurs types de systèmes de fichiers**
  - Spécifiques à un type de périphériques
    - Disquette
    - CD-ROM
    - DVD
    - Bande magnétique
  - Pour un même périphérique (disque dur)
    - ext2, ext3, jfs, ufs, ffs, reiserfs, ...

Création de systèmes de fichiers

- **La création d'un système de fichiers s'effectue avec la commande newfs ou mkfs (dépend de l'OS)**
- **La syntaxe est du type :**
  - mkfs [-t <fstype>] [file_options] device
  - Elle permet de créer un système de fichiers de type *fstype* avec les options *fs_options* sur la partition *device*

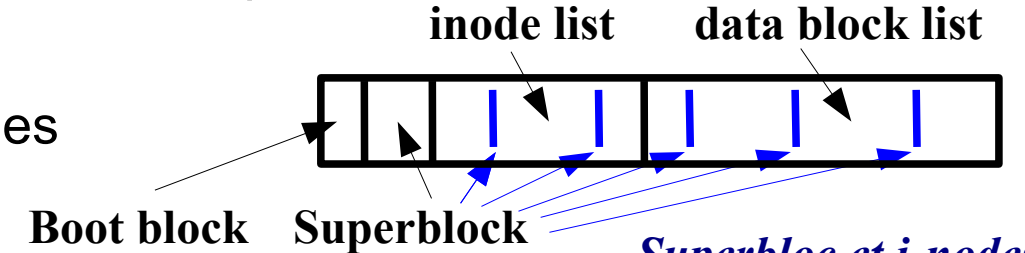
- **Une fois le système de fichiers créé, on peut monter la partition dans l'arborescence avec la commande mount**
- **La syntaxe est du type :**
  - mount [-t <fstype>] [-o <options>] device directory
  - Elle permet de monter la partition *device* de type *fstype* avec les options *options* dans le répertoire *directory* de l'arborescence
  - Le fichier fstab (ou vfstab), que l'on vient de voir, permet de pré-définir le type de système de fichiers, les options de montage et la partition
- **Presque tous les systèmes intègrent des « volume managers » qui sont capables de monter directement des partitions ou des périphériques (volmgt, supermount, automounter, ...)**

- **Mais certains systèmes de fichiers n'ont pas besoin d'être créer**
- **Lesquels ?**

proc	→	N'a pas besoin d'être créé
swap	→	A besoin d'être créé
devfs	→	N'a pas besoin d'être créé
iso9660	→	A besoin d'être créé (mais pas avec mkfs)
vfat	→	A besoin d'être créé (mais pas avec format)

Un système de fichiers, c'est quoi ?

- **Un système de fichiers de type UNIX se compose de :**
  - Un bloc de boot
    - Début de la partition : contient le boot loader
  - Un superbloc
    - Recopié au sein de la partition pour avoir des copies de secours
    - Sa corruption interdit l'utilisation du système de fichiers
    - Il est composé de :
      - Taille du système de fichiers
      - Nombre et liste de blocs libres ainsi qu'un index sur le prochain bloc libre
      - Taille de la liste des i-nodes, nombre d'i-nodes libres ainsi qu'un index sur la prochaine i-node libre
      - Verrous pour les listes de blocs libres et d'i-nodes libres
      - Drapeau de modification du superbloc
  - Une liste d'i-nodes
  - Une liste de blocs de données



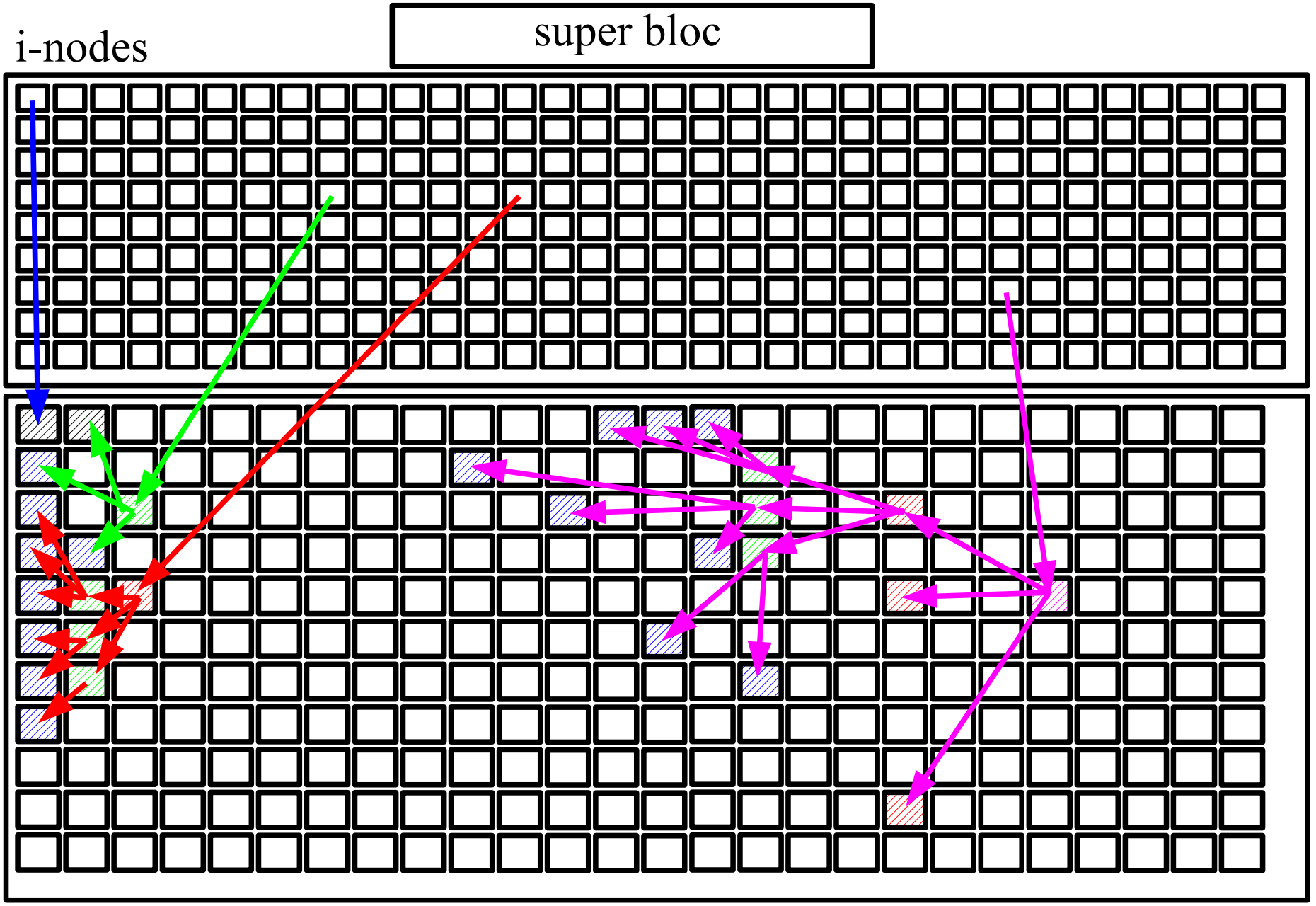
Superbloc et i-nodes

Un système de fichiers, c'est quoi ?

- **Un système de fichiers de type UNIX se compose de :**
  - Un bloc de boot
  - Un superbloc
  - Une liste d'i-nodes
    - une i-node correspond à une entrée dans le système de fichiers
    - une i-node contient :
      - propriétaires du fichier (utilisateur et groupe)
      - type de fichier
      - permissions d'accès
      - dates d'accès (modificication du fichier, accès au fichier, modification de l'i-node)
      - nombre de liens
      - adresse disque d'un bloc de données
      - taille du fichier
  - Une liste de blocs de données
    - un bloc de données contient les données d'un fichier ou une liste d'adresses disque de blocs de données

Superbloc et i-nodes

Un système de fichiers, c'est quoi ?



Superbloc et i-nodes

Garder un système de fichiers consistant

- **Un système de fichiers peut devenir inconsistant → Pourquoi ?**
- **Parce que les systèmes de fichiers sont asynchrones**
- **Pour des raisons de performances, les modifications du système de fichiers sont enregistrées dans un cache en mémoire virtuelle et propagées sur le disque périodiquement (flush disque)**
  - c'est ce que fait la commande sync
- **Comme l'accès au cache est beaucoup plus rapide que celui au disque, la lenteur d'accès au disque dur est atténuée (il n'est pas nécessaire d'attendre la fin d'une opération d'entrée/sortie disque pour rendre la main au processus demandeur)**
- **Problème : le système de fichier « disque + cache » est consistant mais le système de fichier « disque » ne l'est pas forcément à un instant t**
- **Et si à cet instant t, le système crash**
  - Au reboot, le cache est perdu et ne subsiste que le système de fichier sur disque qui n'est plus consistant

- **Mais c'est quoi un système de fichier inconsistant ?**

Pourquoi fsck ?

Garder un système de fichiers consistant

- **Mais c'est quoi un système de fichier inconsistant ?**
- **Exemple de la création d'un nouveau fichier composé de 3 blocs de données**
  - Après la création du fichier, 5 nouveaux blocs sont dans le cache :
    - un nouvel i-node (étape I)
    - un nouveau bloc indirect (étape B)
    - 3 blocs de données (étape D)
  - et 5 blocs ne peuvent être écrits sur le disque en une seule fois (il n'y a pas d'atomicité)

  - Un crash après une étape, quelle qu'elle soit (I, B ou D), conduit à l'inconsistance du système de fichiers
  - De même, un crash après 2 étapes (sur les 3 a réalisé) conduit à l'inconsistance du système de fichiers

- **En conclusion, un système de fichiers inconsistant est un système de fichiers, dans lequel de l'information a été perdue car la liste chaînée, stockant les fichiers, a été rompue**

Pourquoi fsck ?

La commande FSCK

- **La commande FSCK inspecte un système de fichiers pour trouver et, le cas échéant, essayer de réparer les inconsistances**
- **Qu'essaie de détecter FSCK :**
  - les blocs appartenant à un i-node et notés libres
  - les blocs qui ne sont pas notés libres et qui n'appartiennent à aucune i-node
  - les blocs appartenant à plusieurs i-nodes
  - une i-node avec un nombre de liens différents de 0 mais n'appartenant à aucun répertoire
  - une i-node marquée libre mais présente dans un répertoire
  - une i-node corrompue
  - un nombre de liens d'une i-node différent des liens trouvés dans les répertoires
  - le nombre de blocs de données libres du disque n'est pas celui trouvé dans le superbloc
  - le nombre d'i-nodes libre du disque n'est pas celui trouvé dans le superbloc
  - ...

Pourquoi fsck ?

Les autres types de systèmes de fichiers

- **La commande FSCK a un gros inconvénient → c'est LENT !!!**
- **Et pendant que la commande FSCK fait son oeuvre, le serveur n'est pas disponible**
  - Il est dangereux de faire un FSCK sur un système de fichiers montés en lecture/écriture
- **Le reboot peut donc, prendre plusieurs heures sur de gros serveurs**
- **Y-a-t-il un solution à l'utilisation de la commande FSCK lors du reboot d'une machine suite à un crash?**
  - OUI, mais il faut changer le type de systèmes de fichiers afin d'assurer une certaine consistance à n'importe quel instant t
  - Il existe 3 grands types :
    - les systèmes de fichiers « soft updates »
      - plutôt orienté BSD (extension de FFS)
    - les systèmes de fichiers journalisés (ou loggés)
      - ext3, reiserfs, xfs
    - les systèmes de fichiers « log-structured »
      - Peu utilisé (implémentation NETBSD)

Se passer de fsck

Le système de fichiers avec Soft Updates

- **Basé sur FFS (Fast FileSystem)**
- **N'empêche pas l'utilisation de FSCK, mais cette commande sera beaucoup plus rapide**
- **Sans Soft Updates, FFS ne peut assurer sa consistance qu'en mode synchrone (en mode asynchrone, FSCK est obligatoire)**

- **Gestion des méta-data : les données de la structure du système de fichiers (tout sauf les données contenues dans les fichiers)**

- **Principe :**
  - Il faut ordonné les écritures sur disque selon les dépendances qui existent entre les méta-data
  - Dans le but que les seules inconsistances possibles soient la présence de blocs de données et d'i-nodes marquées allouées alors qu'ils sont libres
  - Ce type d'inconsistance n'est pas fatal pour l'utilisation du système de fichiers et la correction peut intervenir en tâche de fond sur un système de fichiers monté en lecture/écriture

Il faut trier

Le système de fichiers avec Soft Updates

- **Exemple : Création d'un nouveau fichier**
  - Pour cela, il faut :
    - Prendre une i-node libre, la renseigner (création de la liste chaînée), écrire les blocs de données → ACTION 1
      - Allocations de nouveaux blocs (i-node + données)
    - Modifier l'i-node du répertoire père pour y intégrer l'i-node du nouveau fichier fraîchement créé → ACTION 2
      - MAJ de blocs (i-node + données) voire allocation de blocs de données
  - Si (ACTION 2, ACTION 1) avec CRASH entre les 2 actions, l'inconsistance ne concerne **pas** que des blocs alloués qui devraient être libres **(entrée d'un répertoire à mettre à jour)**
  - Si (ACTION 1, ACTION 2) avec CRASH entre les 2 actions, l'inconsistance **ne concerne que** des blocs alloués qu'il faut libérer pour avoir un système de fichiers consistant
  - Il faut, donc, créer une dépendance entre l'ACTION 1 et l'ACTION 2 pour que la mise à jour du disque pour l'ACTION 2 ne se réalise que si l'ACTION 1 a été réalisée sur le disque

Il faut trier

Le système de fichiers avec Soft Updates

- **Les dépendances peuvent amener à avoir des cycles (A doit être fait avant B qui dépend de A)**
- **Utilisation de roll-back au niveau des méta-data pour casser ces dépendances**

- **Avantages :**
  - Ne modifie pas la structure du système de fichiers (ajout de fonctions dans le noyau)
  - Moins agressif que la journalisation

- **Inconvénients :**
  - Plus lent lors de l'utilisation (nécessite plus d'écritures qu'un système de fichiers standard)
  - Taille disponible peut être sous-estimée
  - Libération de blocs plus longue

Il faut trier

Le système de fichiers avec journalisation

- **Même principe que les systèmes de gestion de bases de donnée :**
  - REDO LOG
- **La commande FSCK n'est plus utile, c'est la couche noyau du système de fichiers journalisé qui gère les modifications à réaliser pour restaurer un système de fichiers consistant**

- **Gestion des méta-data : les données de la structure du système de fichiers (tout sauf les données contenues dans les fichiers)**

- **Principe :**
  - Pour chaque mise à jour, on utilise le principe de la transaction
  - Dans une transaction :
    - On dit ce que l'on va faire
    - On le fait
    - On dit qu'on l'a fait
  - Les transactions sont consignées dans un fichier de log, utilisé pour recouvrir (recovery) un système de fichier inconsistant

Journalisons

Le système de fichiers avec journalisation

- **Exemple : Création d'un fichier**
  - Dans le log, on écrit :
    - Début de transaction
    - Descripteur de transaction
    - Données à écrire (i-node, données, éventuellement les blocs intermédiaires)
    - Fin de transaction
  - On écrit les données sur le disque (de manière ordonnée ou pas)
  - Les données écrites, on enlève la transaction du fichier de log

- **Recovery :**
  - Si l'entrée du fichier de log n'est pas complète → cette transaction est ignorée et retirée
  - Si une entrée du fichier de log est présente, mais les modifications non réalisées → on les réalise
  - Si les modifications sont présentes sur le disque, mais que la transaction n'a pas été retirée → on la retire et c'est tout

Journalisons

Le système de fichiers avec journalisation

- **Différents modes de journalisation :**
  - Mode data : toutes les données sont journalisées
    - fort impact sur les performances : les données sont écrites deux fois (une fois dans les logs et une nouvelle fois sur le disque)
  - Mode writeback : ne concerne que les méta-data
    - Mais les blocs de données sur le disque peuvent être écrites à n'importe quel moment (seuls les blocs de méta-data sont journalisés)
  - Mode ordered : ne concerne que les méta-data
    - Mais les blocs de données sur le disque sont écrits de manières ordonnées par rapport aux données de transaction

- **Le fichier de log (appelé « write ahead log » ou journal) peut être de deux types :**
  - un fichier spécial présent dans le système de fichiers
  - une partition spécifiquement utilisée pour la journalisation

Le système de fichiers log-structured

- **Type de système de fichiers peu utilisé**
- **La commande FSCK ne sert plus à rien, de par la structure du FS**
- **Principe :**
  - Le système de fichiers n'est qu'un énorme fichier de log
  - Toute modification du système de fichiers (méta-data + données) est loggé (toutes les écritures sont adjacentes, améliorant les performances en écriture)
  - Le système de fichiers est structuré en segment
  - Chaque segment débute par une entrée de log pour assurer la consistance, l'écriture d'un segment n'étant pas une action atomique (approche journalisation)
  - A l'intérieure d'un segment, les blocs sont écrits de manière ordonnée (approche soft updates)
  - Des index (en cache) sont créés pour améliorer la lecture des fichiers
  - De par sa structure, il est simple de revenir à un système de fichiers consistant en analysant la fin du fichier de log
  - La fragmentation du système de fichiers est un problème (résolu par une analyse et une mise à jour en tâche de fond)

Tout n'est que log

Concepts de Base de l'Administration Système

Types de Fichiers

Types de fichiers

- **Sur un système de fichiers Unix, on peut rencontrer 6 types de fichiers (ne pas oublier que sous Unix, TOUT EST FICHIER)**

- **Fichiers standards** **(-)**
  - contient des données
- **Répertoires** **(d)**
  - définitions de fichiers gérant la hiérarchie du système de fichiers
- **Fichiers "device" de /dev**
  - de type bloc (accès direct) (b)
  - de type caractère (accès séquentiel) (c)
- **Fichiers pipe** **(p)**
  - FIFO permettant la communication entre processus
- **Fichiers socket** **(s)**
  - points d'entrée de communications entre processus basé sur les couches réseau
- **Liens** **(l)**
  - pointeurs vers des fichiers, peut être de type souple ou dur

Concepts de Base de l'Administration Système

Boot Loader et Procédure de Boot Matériel

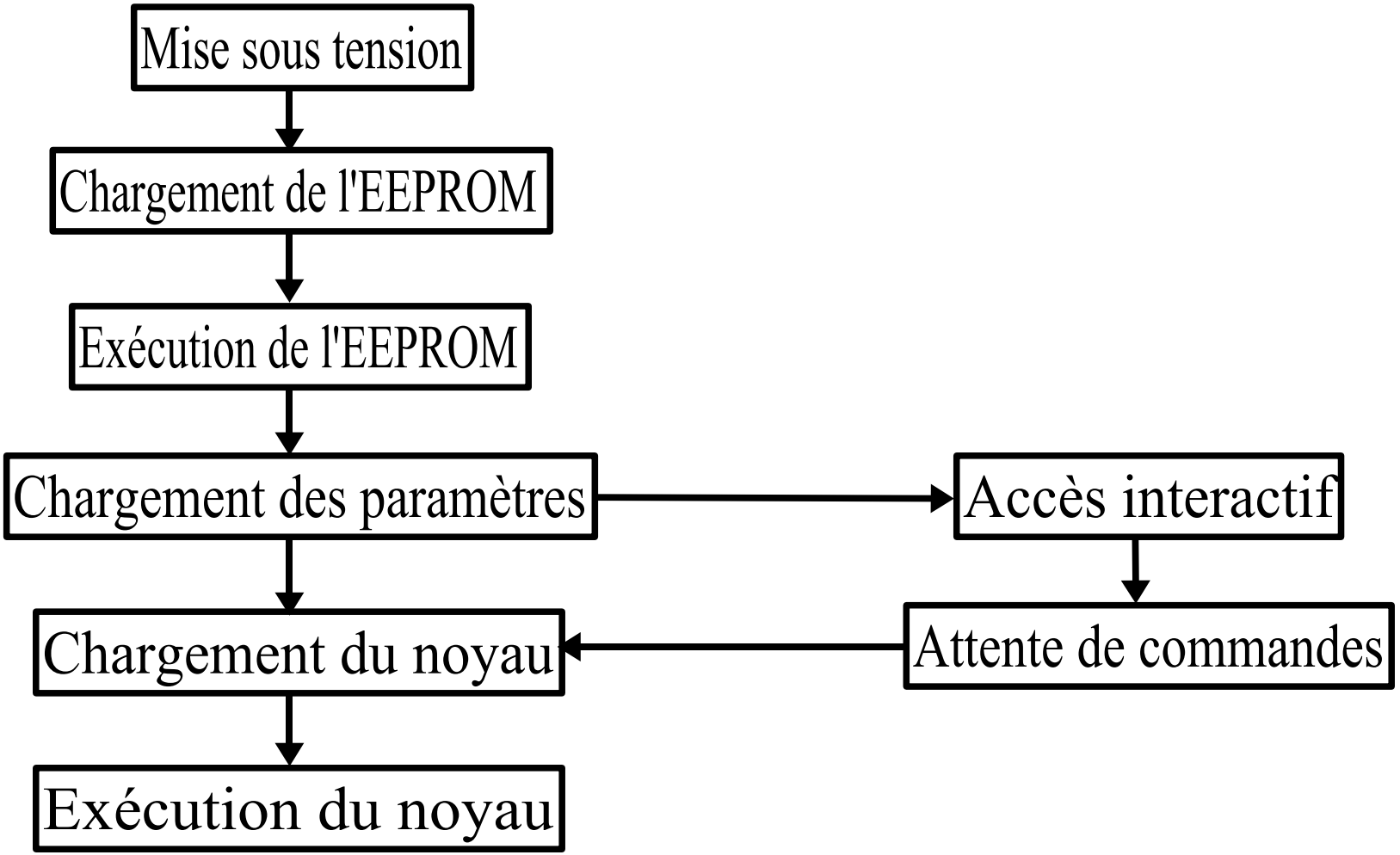
Boot Loader

- **Qu'est-ce qu'un boot loader ?**
  - Pour qu'un système Unix puisse se lancer, il faut que le noyau soit chargé en mémoire et qu'il s'exécute
  - Le but principal d'un boot loader est de chargé le noyau en mémoire et de le lancer
  - Un boot loader peut permettre, aussi, de
    - Choisir entre plusieurs noyaux à charger
    - Passer des paramètres au noyau chargé
    - Choisir entre plusieurs OS (sur un système multi OS)
  - Le boot loader est obligatoire pour charger le noyau
  - Le boot loader fait la transition entre le démarrage matériel de la machine (mise sous tension) et l'exécution du noyau (lancement de l'OS)
  - Le boot loader est dépendant de la plate-forme matérielle
  - 2 types de plates-formes pour exemple :
    - Sun (Solaris)
    - PC (Linux)

Comment ça peut booter ?

- **Sur une SUN**
  - La couche entre le matériel et l'OS se nomme l'EEPROM
  - A la mise sous tension de la station, le mini OS stocké dans l'EEPROM est chargé en mémoire et exécuté
  - L'EEPROM est programmé en forth et donne un « shell » à l'utilisateur
  - Le prompt de l'EEPROM est >>>
  - L'EEPROM permet de définir des variables (stockées en mémoire non volatile) pour définir la façon dont sera chargé le noyau
  - L'EEPROM permet de lister les périphériques de la station et de tester les périphériques
  - L'EEPROM permet de définir le comportement par défaut de la station à sa mise sous tension
  - L'EEPROM va permettre de définir le noyau à charger, le périphérique sur lequel il se trouve et les paramètres à lui passer (grâce aux variables)
  - L'accès à l'EEPROM doit être limité à l'administrateur car sinon une compromission de la station est possible

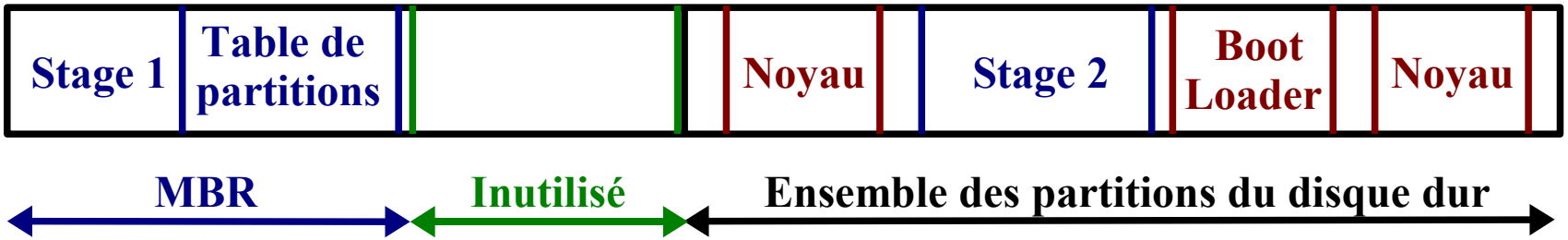
- **En résumé, sur une SUN :**



- **Sur un PC :**
  - Le BIOS est l'équivalent de l'EEPROM
  - Le BIOS gère la machine à la mise sous tension
  - Le BIOS répertorie les périphériques et vérifie qu'ils fonctionnent correctement
    - CPU, mémoire, contrôleurs IDE, disques, lecteurs de disquettes, CD-ROM, clavier, souris, périphériques USB, ...
  - Le BIOS définit un périphérique de boot selon sa configuration et les périphériques détectées
  - Le BIOS charge le MBR (512 octets) en mémoire et l'exécute
  - Si le BIOS ne trouve pas de MBR, il cherche une partition avec le flag bootable et charge les 512 premiers octets de cette partition dans la mémoire et l'exécute
  - Le MBR est un boot loader, mais 512 octets ne sont pas suffisants pour charger un noyau
  - Ces 512 octets ne constituent que le stage 1 du boot loader
  - A partir du stage 1, le boot loader va charger le stage 2 (plus gros) en mémoire et l'exécuter

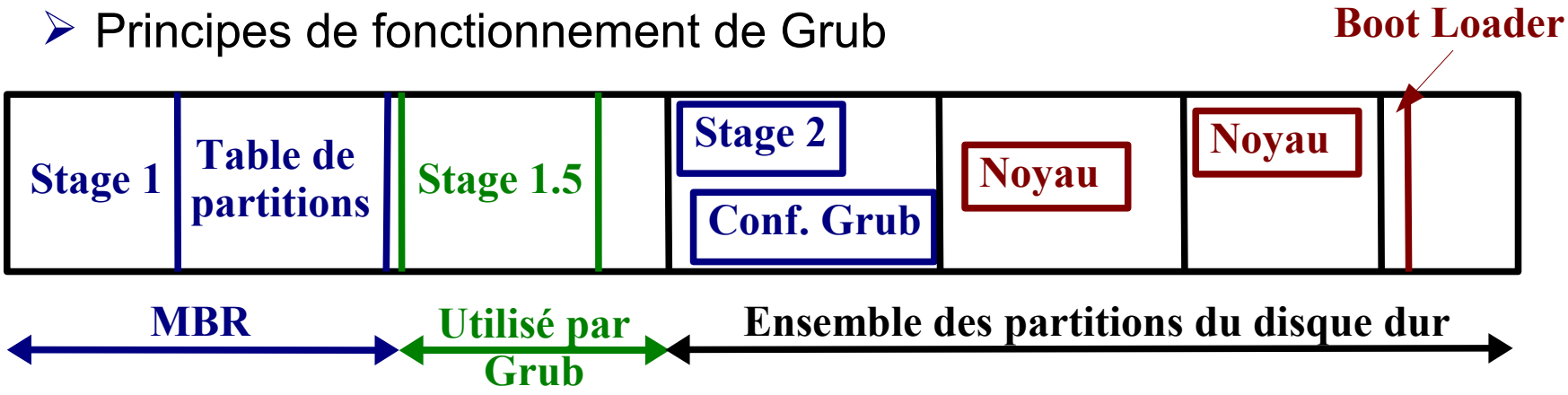
- **Sur un PC :**
  - Le stage 2 va charger le noyau en mémoire et l'exécuter
  - Sous Linux, on trouve 2 boot loaders
    - LILO
      - Premier boot loader de Linux
      - Très flexible
      - Permet de booter autre chose que Linux (Windows, BSD, ...)
      - Ne contient pas de gestion interactive
    - GRUB
      - Boot loader GNU
      - Très flexible
      - Permet de booter autre chose que Linux
      - Possède une gestion interactive
      - Permet la lecture des systèmes de fichiers
    - Pour ces 2 boot loaders, on retrouve les 2 stages qui sont chargés l'un après l'autre

- **Sur un PC :**
  - Principes de fonctionnement de LILO



    - A l'installation de LILO :
      - le stage 2 est créé avec les références statiques aux noyaux pour lesquels il est configuré, ainsi que d'autres boot loaders vers lesquels il peut pointer (*/etc/lilo.conf*)
      - le stage 1 est créé avec les références statiques au stage 2
    - Si l'emplacement du stage 2 est modifié, LILO ne fonctionne plus
    - Si un des emplacements des noyaux est modifié, celui-ci ne peut plus être chargé en mémoire par LILO
    - Donc, pour toute modification (noyau, fichier de configuration) :
      - LILO doit être ré-installé (stage 2 à modifier, qui implique la modification du stage 1)

- **Sur un PC :**
  - Principes de fonctionnement de Grub

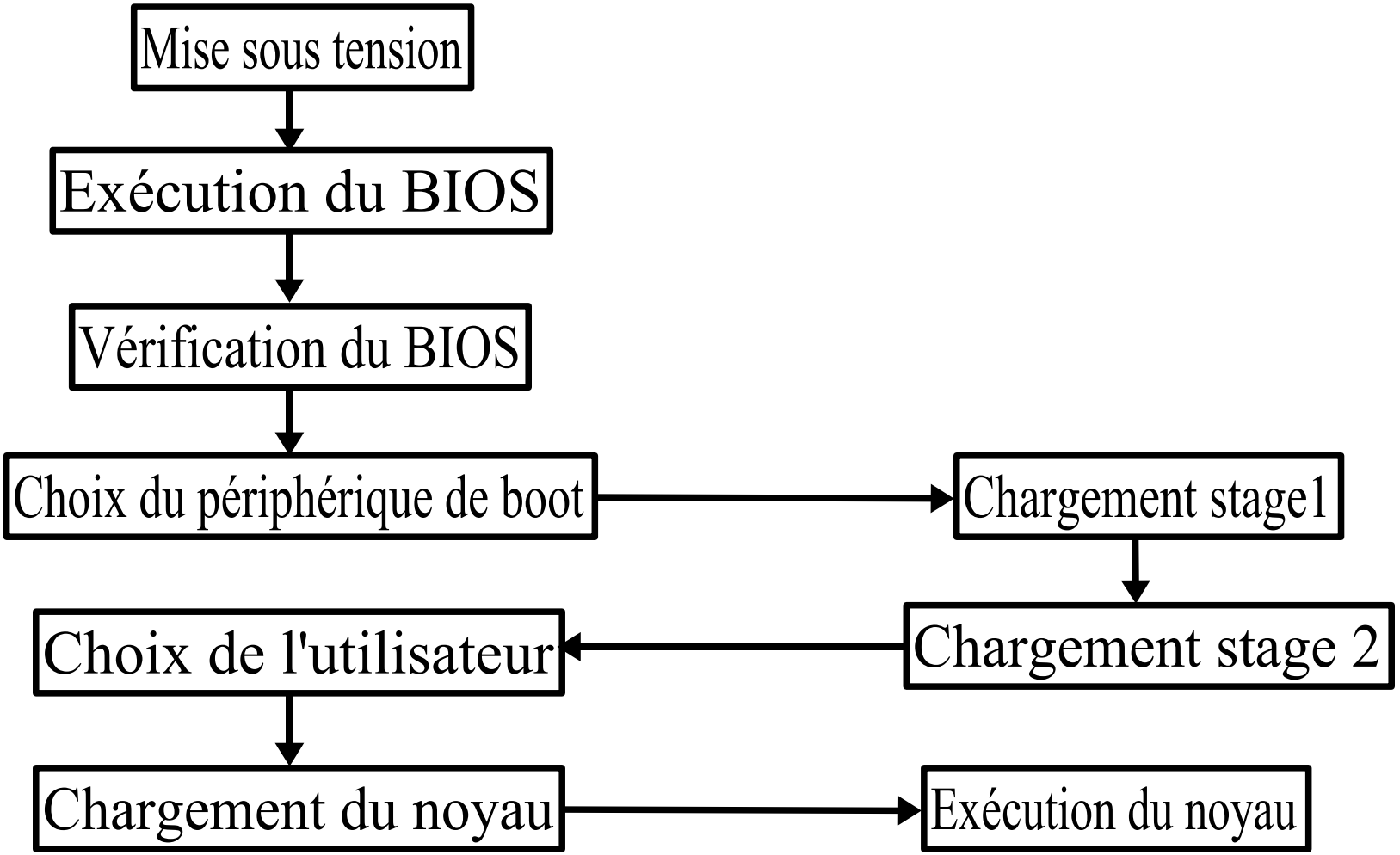


    - A l'installation de Grub, les stages 1, 1.5 et 2 sont copiés
      - Le stage 1 est installé pour charger le stage 1.5
      - Le stage 1.5 permet de lire un système de fichiers spécifique et charge le stage 2 qui est vu comme un fichier
      - Le stage 2 permet de lire le fichier de configuration Grub (*/boot/grub/menu.lst*)
    - Les noyaux à charger en mémoire sont vus comme des fichiers, en cas de modification, pas besoin de ré-installer Grub
    - Idem pour les boot loaders qui sont vu comme des partitions
    - Idem pour le fichier de configuration de Grub
    - Alors, dans quel(s) cas faut-il ré-installer Grub ?

Boot Loader et BIOS

- **Sur un PC :**
  - Sous OpenBSD, le boot loader est stocké sur la partition DOS, dans les 512 premiers octets
  - Ces 512 premiers octets composent le stage 1
  - Le stage 1, comme pour Linux, charge le stage 2 qui se trouve sur le disque, dans la partition racine
  - Le stage 2 permet un accès interactif au boot loader à la manière de l'EEPROM de SUN
  - Cet accès interactif permet de choisir le noyau, de pré-configurer le noyau et de passer des paramètres aux noyau (single user, par exemple)
  - Le principe reste le même : charger le noyau et l'exécuter

- **En résumé sur un PC**

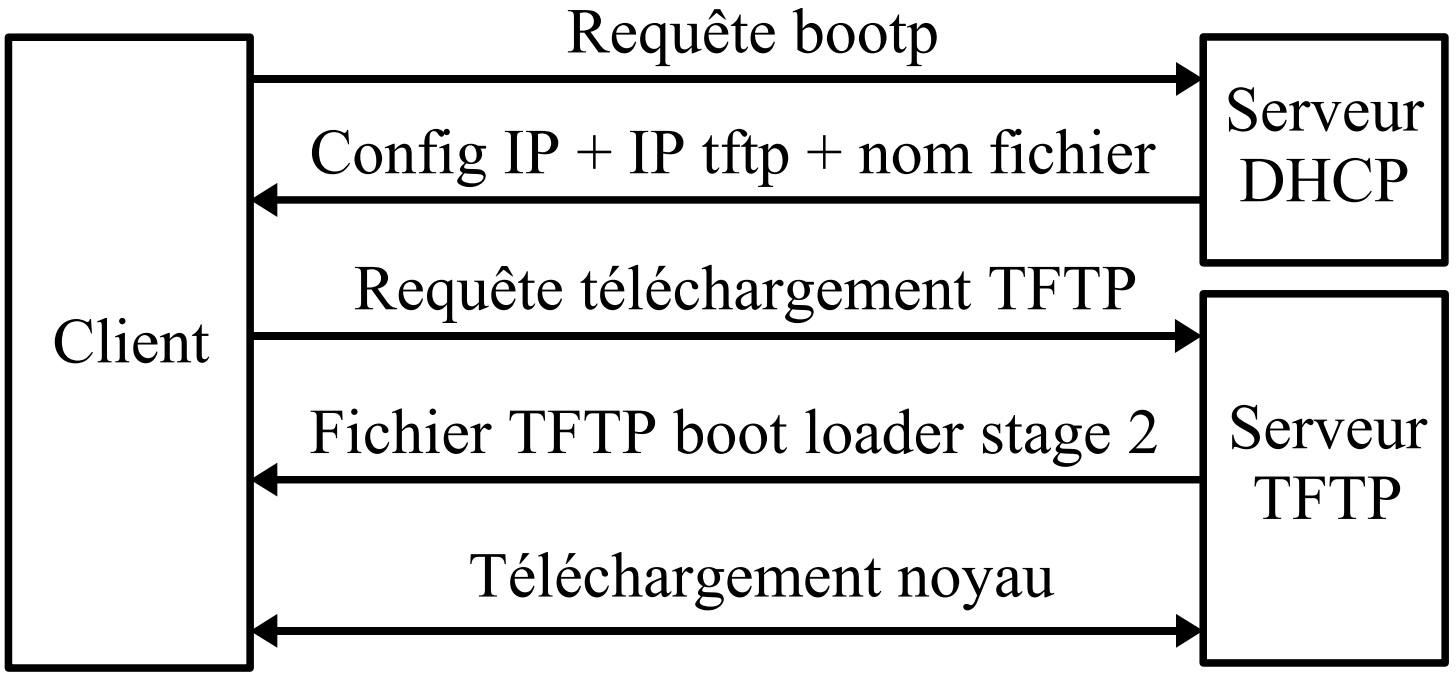


- **Sur SUN et sur PC, il existe d'autres méthodes de boot**
  - Par le réseau
  - Sur CD-ROM
  - Sur bandes de types DAT ou DDS

- **Pour le réseau, comment ça marche ?**
  - Sur SUN
    - Toutes les machines SUN sont capables de booter par le réseau
    - Au boot, la carte réseau envoie un broadcast bootp
    - Un serveur DHCP renvoie la configuration IP de la machine et une référence à un fichier de boot (adresse IP d'un serveur TFTP et un nom de fichier)
    - Une fois que la carte réseau a une adresse IP, elle contacte le serveur TFTP et télécharge le fichier de boot
    - Ce fichier est chargé en mémoire et exécuté, il s'agit du noyau
    - Cette méthode est très utilisée pour l'installation de stations à travers le réseau en s'appuyant sur des systèmes de fichiers réseau de type NFS

Boot Loader et réseau

- **Pour le réseau, comment ça marche ?**
  - Sur PC
    - Seuls les PCs possédant une carte réseau compatible PXE sont capables de booter par le réseau
    - Une carte réseau PXE est sélectionnable par le BIOS comme périphérique de boot
    - Le principe de fonctionnement est le même que pour les SUNs



- **Le boot sur CD-ROM**
  - La norme des systèmes de fichiers sur CD-ROM est ISO9660
  - Cette norme comporte plusieurs extensions
    - Joliet : Noms longs type FAT
    - Rock-ridge : Extensions type Unix (protections, propriétaire, ...)
    - El-Torito : CD-ROM bootable

- **Pour les stations SUN**
  - Les CD-ROM bootables sont vus comme des disques multi-partitions de type BSD
  - Le boot sur un CD-ROM est très similaires au boot sur un disque dur

- **Pour les Pcs**
  - Le CD-ROM El-Torito est obligatoire
  - Un CD-ROM El-Torito simule une disquette
  - Le CD-ROM contient un fichier qui est l'image d'une disquette bit à bit
  - C'est cette image qui est utilisée pour booter

Boot Loader

- **Le boot sur CD-ROM est très important pour plusieurs raisons**
  - Permet l'installation de l'OS
  - Permet le boot sans utiliser le disque dur
    - Pour réparer un système endommagé
    - Pour sauvegarder un système
    - Pour upgrader un système
    - ATTENTION : permet, aussi, de pirater le système
  - Permet l'utilisation de systèmes sécurisés
    - Le système de fichiers est en mémoire RAM et/ou sur le CD-ROM, sans modification possible
- **Le boot sur bandes est possibles sur des matériels spécifiques (HP, par exemple)**
  - Cela permet, après création de bandes spécifiques, de pouvoir booter en single user, pour sauvegarder, réparer ou upgrader

Concepts de Base de l'Administration Système

Démarrage et Arrêt des Systèmes Unix

Démarrage des Systèmes Unix

- Nous venons de voir ce qui se passe de la mise sous tension à l'exécution du noyau
- Mais que se passe-t-il après ? C'est ce que nous allons voir

- Le noyau s'exécute et commence par détecter tous les périphériques de la machine
- En détectant les périphériques, il charge les différents drivers (pilotes)
- Ceci fait, il monte la partition root (passé en paramètre du noyau ou définit en dur dans celui-ci) en lecture seule
- S'il ne peut pas monter de partition root, il lancera son fameux cri du kernel panic
- Lorsque la partition root est montée, il cherche l'exécutable /sbin/init et l'exécute
- C'est init qui terminera la procédure de démarrage
- Init sera toujours présent sur le système en tant que processus, puisqu'il est le processus père de tous les autres processus du système

Après le noyau ...

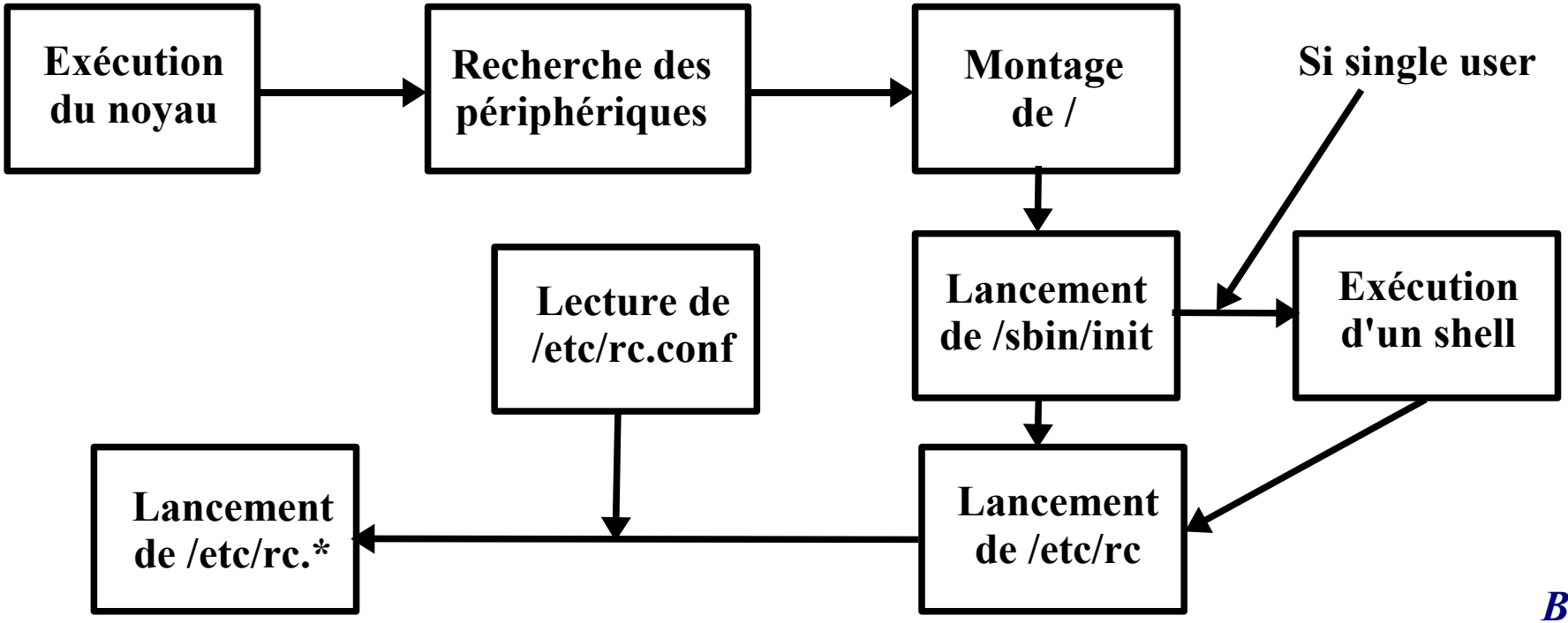
- **Init terminera la procédure de démarrage selon qu'il est BSD ou SYSV**
- **Pour les BSD**
  - init cherche le script /etc/rc et l'exécute
  - Le script /etc/rc va exécuter d'autres scripts dans /etc et commençant par rc (rc.boot, rc.local, rc.net, ...)
  - Le script /etc/rc et les scripts lancés utilise le fichier /etc/rc.conf qui regroupe les variables définissant le comportement des scripts
  - Exemple :
    - La variable SMTP définira le lancement d'un MTA
      - Si cette variable est à YES
      - Si cette variable n'est pas à YES, le MTA ne sera pas lancé
  - Le comportement du démarrage est totalement paramétrable grâce au fichier /etc/rc.conf
  - Les différents scripts rc.* réalisent des actions de base nécessaires au démarrage

Facile sous BSD

- **Pour les BSD**
  - Les différentes actions des scripts rc.* sont :
    - Vérifier les systèmes de fichiers
    - Monter les systèmes de fichiers
    - Lancer les différents démons et services
    - Lancer les bannières de connexion
    - Configurer le réseau
    - Lancer l'interface graphique, ...
  - Les systèmes BSD possèdent deux niveaux de fonctionnement :
    - Single user et multi-user
  - Le niveau multi-user est le mode de fonctionnement normal tel que défini plus haut
  - Le niveau single user est choisi par l'utilisateur lors du boot (option -s)
  - Cette option est passée par le noyau à init qui lance un shell au lieu de /etc/rc
  - A la sortie de ce shell, init lancera /etc/rc et le système démarrera en mode multi-user

- **Pour les BSD**
  - Le mode single user permet de booter sur un système minimal (sans scripts de démarrage)
    - Pour réparer un système
    - Pour sauvegarder un système
    - Pour mettre à jour un système

- **En résumé**



BSD

- **Pour les SYSV**
  - C'est à la fois plus simple et plus compliqué
  - Comme pour BSD, le noyau lance /sbin/init après avoir monté /
  - Mais, à la différence de BSD, init lit un fichier de configuration : /etc/inittab
  - Ce fichier de configuration définit des niveaux d'exécution (run-levels)
  - Les niveaux d'exécution sont :
    - 0 : Pour arrêter le système
    - 1 ou s : Pour passer en mode single user
    - 2 : Mode multi-utilisateurs (exemple : sans réseau)
    - 3 : Mode multi-utilisateurs (exemple : avec réseau)
    - 4 : Mode multi-utilisateurs (exemple : avec NIS)
    - 5 : Mode multi-utilisateurs (exemple : avec X11)
    - 6 : Pour rebooter le système
  - Le fichier inittab définit le run-level par défaut à utiliser au boot
  - Le fichier inittab a une syntaxe assez simple : pour un run-level, on définit les scripts à exécuter

Compliqué sous SYSV

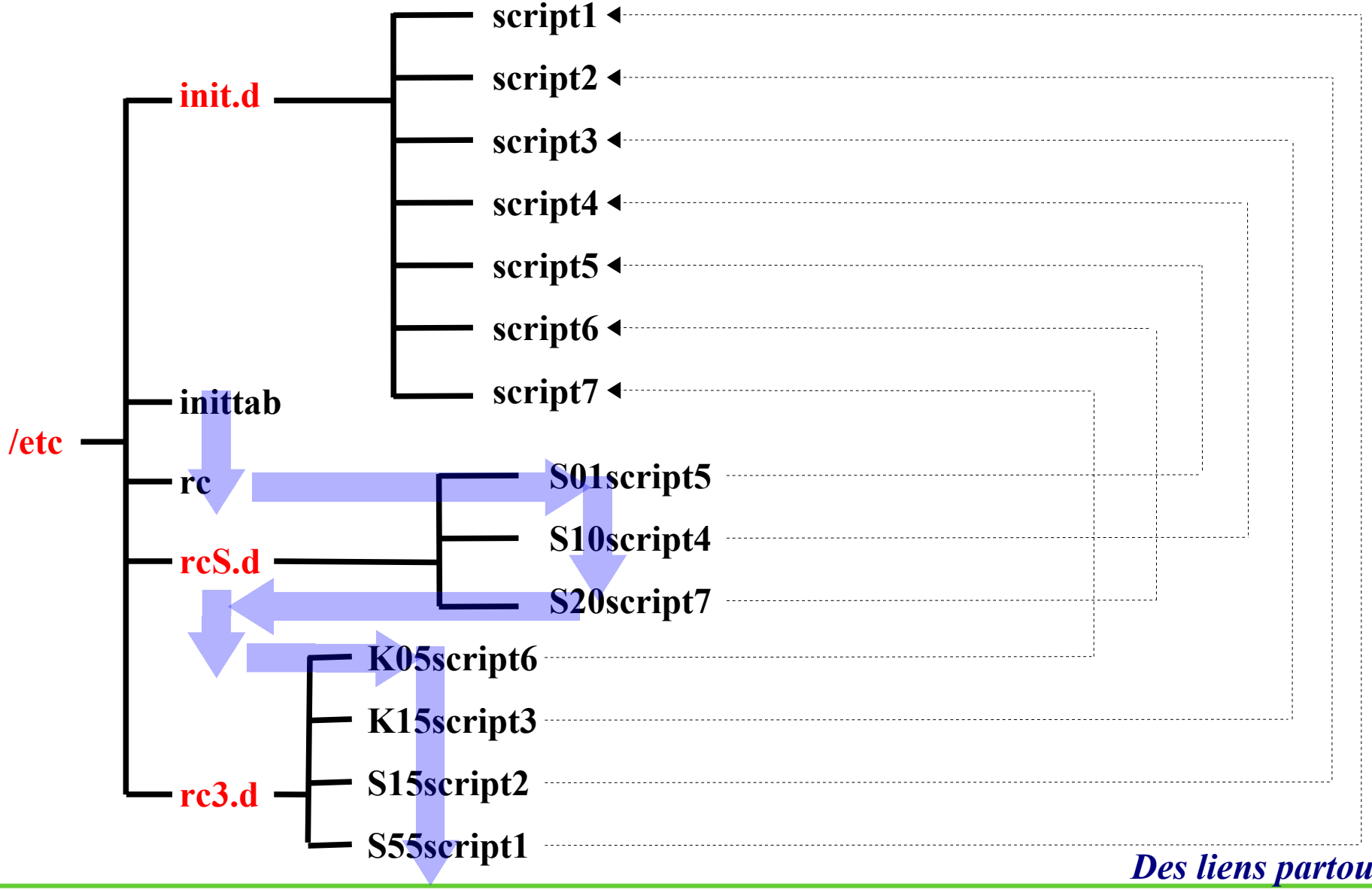
- **Pour les SYSV**
  - Le plus souvent, quelque soit le run-level, inittab lance le même script avec, en paramètre, le numéro du run-level (ou le script rc*n*)
  - Ce script est /etc/init.d/rc
  - Ce script va exécuter les scripts présents dans /etc/rc*n*.d où *n* est le run-level
  - Le répertoire /etc/rc*n*.d contient deux types de fichiers (liens)
    - K*nn** : scripts d'arrêt (Kill)
    - S*nn** : scripts de démarrage (Start)
  - Le nombre *nn* permet d'ordonner le séquencement de ce qui sera lancé
  - Les scripts K*nn** et S*nn** sont des liens (souples ou durs selon les Unix) vers les vrais scripts dans /etc/init.d
  - Ceci permet, en supprimant le lien, de ne plus lancer le script pour le run-level donné, mais de ne pas effacer le script (idem pour rajouter un nouveau script, il suffit de créer un lien)
  - Un script K*nn** est lancé par le script rc avec le paramètre **stop**
  - Pour S*nn**, le script rc utilise le paramètre **start**

Usine à gaz

- **Pour les SYSV**
  - Un seul script est présent dans /etc/init.d, à la fois pour le démarrage et l'arrêt d'un service, mais il doit prendre en argument **start** ou **stop**
  - Selon les Unix, l'ordre entre K et S est différent, il faut le vérifier dans les scripts
  - Exemple : On passe du run-level 3 vers le run-level 5
    - rc exécute les K*nn** dans /etc/rc3.d
    - Puis rc exécute les S*nn** dans /etc/rc5.d
    - Sur d'autres systèmes, rc exécute les K*nn** dans /etc/rc5.d puis les S*nn** dans /etc/rc5.d
  - Souvent, on trouve un script rc.boot ou un répertoire /etc/rcS.d, qui permet l'exécution des scripts spécifiques du boot de la machine (remontage de / en lecture/écriture, vérification des systèmes de fichiers, montage des partitions de swap, ...)
  - Ce script est exécuté dans un run-level spécifique de inittab, définissant le boot de la machine

Usine à gaz

- **Pour les SYSV**
  - En résumé



Des liens partout

- **Pour les SYSV**
  - Certains systèmes SYSV combinent les deux méthodes (SYSV et BSD)
  - Comment ?
  - Ils sont SYSV, donc, ils utilisent les run-levels et /etc/inittab
  - Mais pour chaque run-level, un script seul est exécuté, sans utilisation du mécanisme de répertoire /etc/rc*n*.d et les liens vers /etc/init.d
  - Exemples :
    - Linux Slackware
    - Linux SuSE (avec /etc/rc.conf)
    - Mais Linux/Debian est totalement SYSV

Et si on mélangeait

- **D'autres systèmes de démarrage sont en train de voir le jour**
  - Basés sur les daemontools de D.J. Bernstein
    - Permet de gérer les services Unix (daemon)
    - Chaque service Unix à lancer est supervisé par un processus des daemontools
    - Si le service s'arrête, le superviseur le relancera
    - Une commande permet de vérifier que les superviseurs sont OK
    - Ne remplace pas `init` qui devra lancer les daemontools au démarrage

Mais les choses évoluent

- **D'autres systèmes de démarrage sont en train de voir le jour**
  - Upstart d'Ubuntu
    - Remplace purement et simplement `init`
    - Intègre la notion de dépendance
    - Orienté événements
    - Un job de upstart peut émuler le fonctionnement SYSV (ou BSD) pour migrer progressivement

Mais les choses évoluent

Arrêt des Systèmes Unix

- **Pourquoi faut-il arrêter un système Unix proprement ?**
- **Pour 2 raisons principales :**
  - Pour arrêter les processus qui fonctionnent et libérer les différentes ressources utilisées
    - Mémoire
    - Connexions réseau
    - Fichiers disques ouverts
  - Mais, surtout, pour éviter la corruption des systèmes de fichiers
    - Unix utilise la méthode asynchrone d'écriture sur disque
    - Donc, lorsqu'un fichier est modifié, cette modification ne sera effective sur le disque qu'au bout d'un certain temps
    - Si le système de fichiers n'est pas synchronisé (sync) avant l'arrêt, le système de fichiers peut être corrompu
    - Il devra être vérifié et réparé (fsck) au redémarrage, lors du montage du système de fichiers, ce qui peut prendre beaucoup de temps
    - Sans compter la perte possible d'informations !!!

Arrêter tout

Arrêt des Systèmes Unix

- **Comment arrêter un système Unix ?**
- **Pour SYSV, en utilisant les run-levels 0 ou 6**
- **Pour BSD, en utilisant les commandes qui vont permettre l'exécution du script /etc/rc.shutdown par init et permettre le reboot ou l'arrêt**

- **Plusieurs commandes sont disponibles**

  - `Shutdown` (commande à préférer)
    - Permet de changer le run-level vers 0, 1 ou 6, en demandant ce changement de run-level à `init` (SYSV)
    - Demande à `init` d'exécuter `/etc/rc.shutdown`, puis `reboot` ou arrête la station (BSD)

  - `Halt`
    - Arrête le système en urgence (sans passer par `init`) en synchronisant les systèmes de fichiers et en arrêtant la station

Arrêter tout

- **Plusieurs commandes sont disponibles**
  - `Reboot`
    - Arrête le système en urgence (sans passer par `init`) en synchronisant les systèmes de fichiers et en rebootant la station
    - Certaines implémentations permettent de force le reboot en single user
  - `Telinit`
    - Permet d'envoyer un message à `init` pour modifier le run-level et exécuter les scripts correspondants
    - Permet, par exemple, de passer en single user sans rebooter la station

Arrêter tout

Concepts de Base de l'Administration Système

Démons et Lancement de Services

Démons et Lancement de Services

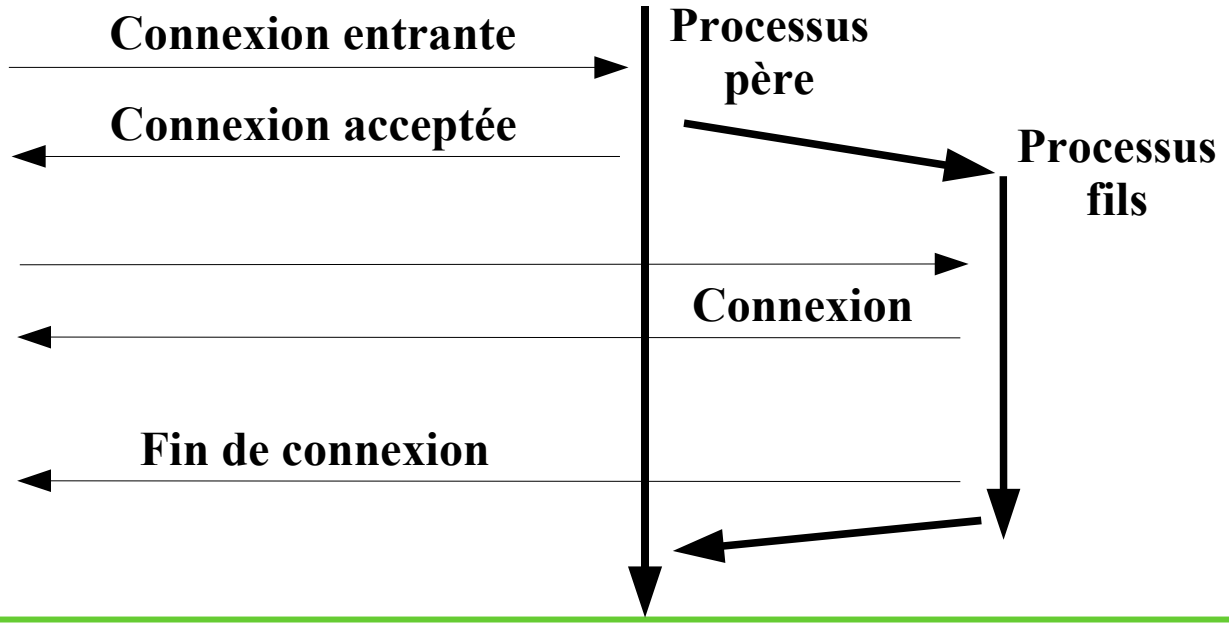
- **Nous venons de voir comment les systèmes Unix bootent et démarrent**
- **Lors de ce démarrage, ils exécutent des scripts qui configurent la machine et lancent des démons**
- **Mais que sont les démons ?**

- **Le mot « démon » vient de « daemon » qui, en anglais, désigne une divinité immortelle (et non un démon au sens « devil »)**
- **Un démon est un processus qui ne meurt jamais, qui ne s'arrête jamais**
- **Les démons sont les services qui sont toujours présents dans la liste des processus (ps)**

- **On trouve deux types de démons :**
  - Les démons de type réseau (les plus nombreux) : apache, portmapper, sshd, inetd, ...
  - Les démons non réseau : cron, at, syslog, lpd, apm, cardmgr, ...

Les Démons sont parmi nous

- **Le principe de fonctionnement des démons réseau est souvent le même**
  - Un processus fonctionne et écoute un port réseau
  - Lorsque une connexion réseau arrive, le processus père crée un processus fils qui s'occupera de cette connexion jusqu'à ce qu'elle se termine
  - A la fin de la connexion, le processus fils meurt
  - Grâce à cette façon de fonctionner, il existe toujours un processus (le processus père) pour gérer une nouvelle connexion entrante



Comment ça marche ?

- **Comment sont lancés ces services ?**

- **Grâce aux scripts de démarrage**
  - Ces scripts sont dans /etc/init.d (SYSV)
  - Ou dans /etc intégrés au scripts globaux rc.* (BSD)

- **Grâce au super démon inetd ou xinetd**
  - Qu'est-ce que inetd (et xinetd, son successeur) ?
  - C'est un processus pour les services réseau
  - Il écoute plusieurs ports et lance les différents serveurs associés à ces ports
  - Intérêts
    - Seuls les processus réellement nécessaires sont présents
    - Permet d'ajouter des paramètres spécifiques pour certains serveurs
    - Permet d'insérer un TCP-Wrapper pour filtrer par adresses IP
  - Mais certains services ne peuvent être gérés par inetd (apache, sshd, portmap, named, yp, ...)

Inetd et xinetd

- **Qu'apporte xinetd par rapport à inetd ?**
  - Des nouvelles fonctionnalités
    - TCP-Wrapper intégré
    - Gestion du chroot
    - Limitation du nombre de serveurs lancés par service
  - Une configuration par répertoires et fichier

- **Comment voir les services lancés ?**
  - Un service est un processus, c'est, donc, avec *ps* que l'on verra les processus du système
  - Pour un service réseau, on pourra utiliser *netstat* pour vérifier les connexions réseau ouvertes, ainsi que les ports ouverts

- **Comment arrêter les services ?**
  - En utilisant les scripts d'arrêt
  - En utilisant la commande *kill* pour arrêter un processus

Concepts de Base de l'Administration Système

Processus, Mémoire Virtuelle et Entrées/Sorties

Introduction au Tuning

- **Qu'est-ce que le tuning ?**
  - C'est, le plus souvent, pouvoir répondre à la question :
    - « Pourquoi le système est-il si lent ? »
  - En fait, c'est étudier l'affectation des ressources d'un système ...
  - ... dans le but, de l'optimiser

  - Le tuning consiste, donc, à paramétrer au mieux un système pour pouvoir l'utiliser au mieux de ses performances
  - Un autre aspect du tuning est l'achat de matériels ou la mise à niveau matériel pour améliorer les performances

- **Les performances du système dépendent de l'affectation des ressources**
- **Les ressources intervenant dans les performances :**
  - CPU
  - Mémoire
  - Entrées/Sorties Disques
  - Réseau et Périphériques

Trouver la bonne fréquence

Introduction au Tuning

- **Problème de performance = insuffisance de ressources**
- **Une insuffisance de ressources ne peut se régler que de 2 façons :**
  - Ajouter des ressources (achat ou MAJ de matériels)
  - Rationner les ressources (tuner le système)
- **Les mécanismes de contrôles des ressources système**
  - CPU
    - Gestion des priorités
    - Traitement par lot et files d'attente
    - Ordonnancement des processus
  - Mémoire
    - Architecture du swap
    - Limitation des ressources utilisées
    - Paramètres de la gestion de la mémoire
  - Entrées/Sorties Disques
    - Architecture du système de fichiers (disques, contrôleurs, ...)
    - Placement des fichiers sur les disques
    - Paramètres des Entrées/Sorties

Trouver la bonne fréquence

Introduction au Tuning

- **Une bonne approche pour gérer des problèmes de performances :**
  - Poser le problème de manière la plus détaillée
  - Déterminer la ou les causes du problème
  - Formuler les améliorations des performances à atteindre
  - Concevoir et implémenter les modifications au niveau du système et des applications
  - Surveiller le système pour vérifier si les modifications ont fonctionné
  - Et recommencer

- **Dans tout problème de performances**
  - Il faut savoir ce qu'il se passe sur son système, en temps normal
  - Nécessite de surveiller le système ...
  - ... et de savoir quand le système dévie ou se comporte anormalement
  - Permet, également, d'avoir un historique du comportement du système

- **Pour chaque ressource, voyons comment la surveiller**

Trouver la bonne fréquence

- **Surveiller le CPU**
  - C'est, en fait, surveiller les processus
  - *uptime* donne des valeurs moyennes de charge
  - Charge : nombre moyen de processus actifs
    - Si > 3, souvent problématique sur un système interactif
  - *ps* donne la liste des processus du système et le pourcentage de CPU qu'ils prennent
  - *top* permet d'avoir la liste des processus en temps réel, ainsi que des informations importantes sur la charge du système
  - *pstree* permet de créer l'arborescence des processus (en intégrant les liens de parentés)
  - *vmstat* donne des informations sur l'utilisation CPU (mode utilisateur, mode noyau et idle)
  - *ps* peut montrer les threads noyau, qu'il ne faut pas confondre avec des processus
  - Sur les systèmes multi-threadés, il ne faut pas confondre processus et threads (un thread est un composant d'un processus et le scheduler système ne voit que les processus)

CPU overclocking

- **Surveiller le CPU**
  - Surveiller les entrées/sorties des processus
    - Quels sont les fichiers ouverts par les processus ?
      - répertoires, fichiers, fichiers temporaires, exécutables, librairies, devices, sockets réseau, pipes, ...
    - *lsof* ou *fuser* sont des commandes Linux permettant de voir quels fichiers sont ouverts par un processus
    - *fstat* est l'équivalent sous FreeBSD et OpenBSD
  - Surveiller les appels système d'un processus
    - Permet une surveillance très fine (trop fine !)
    - Une bonne connaissance du fonctionnement de l'OS est indispensable pour interpréter les informations recueillies
    - Très utile pour déboguer dans, presque, toutes les situations
    - *strace* est la commande Linux et FreeBSD de visualisation des appels système
    - *ktrace* est l'équivalent pour OpenBSD

Surveiller le CPU

- On peut limiter les ressources d'un processus
  - Temps CPU utilisé
  - Taille maximale du segment de données
  - Taille maximale de segment de pile
  - Taille maximale du fichier core
  - Quantité de mémoire virtuelle utilisée
- Toutes ces limitations peuvent être définies par les commandes *limit* ou *ulimit*
- Mais ces limitations ne s'appliquent qu'aux processus
  - Pas de notion d'une connexion utilisateurs
  - Pas de notion de limitations par utilisateur
  - Donc peu utilisable

CPU overclocking

- **Lors de problèmes de performance CPU, 3 solutions peuvent être utilisées :**
  - Gérer les priorités des processus
    - Chaque processus possède deux priorités
      - Priorité de base (définie à la création du processus)
      - Priorité courante (calculée pour allouer le CPU)
    - Commandes *nice* ou *renice* (modifier la priorité de base)
    - Un processus peut être arrêté ou suspendu en lui envoyant un signal : commandes *kill* et *killall*
    - Certains processus ont du mal à mourir
      - Zombis
      - Attente de ressources réseau (type NFS)
      - Attente de ressources E/S (disque ou bandes)

- **Lors de problèmes de performance CPU, 3 solutions peuvent être utilisées :**
  - Déplacer les tâches consommatrices de CPU vers d'autres systèmes ou les exécuter lorsque le système est moins chargée (batch)
    - Utilisation de *cron* (exécution périodique)
    - Utilisation de *at* ou *batch* (exécution différée)
  - Modifier les paramètres d'ordonnancement pour privilégier certains processus
    - Très compliqué
    - Et surtout très risqué pour la stabilité du système
    - A faire, si l'on sait EXACTEMENT ce que l'on fait

CPU overclocking

- **Surveiller la mémoire**
  - Les Unix utilisent des mécanismes de pagination (gestion de l'espace mémoire d'un processus)
  - Ne pas confondre :
    - Pagination (allocation d'une unité de mémoire virtuelle)
    - Swap (transfert d'un processus entier de/vers la zone de stockage secondaire)
    - Trashing (système ne possédant plus assez de mémoire virtuelle pour fonctionner)
  - *vmstat* permet de surveiller l'utilisation de la mémoire
  - *ps* donne la liste des processus avec le pourcentage de mémoire utilisée
  - *top* donne la liste des processus en temps réel avec le pourcentage de mémoire utilisée
  - *free* (sous linux) donne un bon aperçu de l'utilisation globale de la mémoire

J'ai la mémoire qui flanche

- **Surveiller la mémoire**
  - L'espace de pagination (swap) est important
  - Déterminer la taille adéquate pour le swap
    - Difficile
    - Dépend de ce que fait le système
    - Dépend de la configuration globale matérielle du système
  - L'espace de pagination peut être
    - Une partition dédiée (à préférer pour de meilleure performance)
    - Un fichier
  - Penser à bien architecturer l'espace de pagination
    - Répartir sur plusieurs disques
    - Ne pas créer de goulots d'étranglement

J'ai la mémoire qui flanche

- **Architecture des E/S disques**
  - Pour optimiser les E/S, utiliser l'arborescence UNIX
    - */* et */usr* sont utilisés en parallèle
      - Il est judicieux de placer les deux partitions sur deux disques différents sur des contrôleurs différents
    - */var* est plutôt utilisé à la fois en lecture et en écriture
      - Dans le cas d'un serveur, on pourra placer */var* sur un disque dédié
  - L'utilisation de RAID (même logiciel) peut améliorer les performances
  - La fragmentation des fichiers diminue les performances
  - L'utilisation de types de systèmes de fichiers peut améliorer les performances
    - Ext2 est très rapide mais peu robuste en cas de crash
    - Ext3 est très robuste mais moins rapide que ext2
    - Reiserfs est moins rapide que ext3
    - Le paramétrage du système de fichiers peut avoir un impact (taille des clusters, taille réservée à root)

J'ai mes entrées ...

- **Gérer l'espace disque**
  - Les commandes *du* et *df* permettent de surveiller l'espace disque utilisé
    - *df* donne des informations sur les systèmes de fichiers
    - *du* donne la taille d'un répertoire sur disque
    - *quot* permet d'avoir la consommation d'espace disque par utilisateur
  - Définir les fichiers inutiles (scripts automatiques)
  - Définir une politique pour les fichiers inutiles
  - Compresser les fichiers peut utilisés
  - Convaincre les utilisateurs de faire le ménage
  - Proposer des outils d'archivage de fichiers transparents pour l'utilisateur
  - Définir une politique pour les fichiers non accédés depuis un certain temps :
    - Archivage (disque, CD-ROM, bandes)
    - Récupération (transparente si possible)

Ne pas saturer un disque

- **Gérer l'espace disque**
  - Limiter la taille des fichiers de log
    - Utilisation de logrotate
    - Limitation des fichiers core
    - Surveiller */var* et le purger le cas échéant
  - L'utilisation des quotas
    - Permet de limiter l'espace utilisé pour un utilisateur
    - Permet de limiter l'espace utilisé pour un groupe d'utilisateur

Ne pas saturer un disque

Exploitation d'un système Unix

Seconde Partie du Cours

PLAN (rappel)

- **Exploitation d'un système Unix**
  - Les comptes utilisateurs (ou comment autoriser l'utilisation du système)
  - L'authentification (password, groups, shadow et PAM)
  - Les permissions sur les fichiers, les quotas disque et les ACLs (ou comment contrôler qui accède à quoi)
  - Exécution décalée : cron, at et les scripts d'exploitation (ou comment automatiser son travail)
  - Le noyau : fonctionnement, modules, configuration et compilation (ou comment empêcher sa station de booter)
  - X-Window (ou comment profiter d'une interface graphique)
  - Sauvegardes et restaurations (ou comment prendre une assurance pour son système)
  - Les impressions (ou comment s'amuser de longues heures avec les imprimantes)
  - Le réseau (ou comment configurer les services de base pour communiquer avec le monde entier)
  - Syslog et Accounting (ou comment jouer à Big Brother)

Exploitation d'un système Unix

Gestion des Comptes Utilisateur, Authentification, PAM, SU et SUDO

Identification et Authentification

- **Identification**
  - Savoir qui est qui, pour déterminer les accès autorisés
  - Notion de login ou username

- **Authentification**
  - Prouver qui on prétend être
  - Notion de secret partagé ou password

J'crois qu'j'ai mon compte

UID et GID

- **UID : User IDentifier**
  - Identifie de manière unique un utilisateur sur le système
  - Nous sommes tous des numéros
  - UIDs spécifiques
    - 0 : ROOT
    - Notions d'utilisateurs spécifiques pour gérer la séparation des privilèges des daemons (lp, mail, sshd, ...)
- **GID : Group IDentifier**
  - Notion de groupes d'utilisateurs
  - Tout utilisateur appartient à au moins un groupe
  - Un utilisateur peut appartenir à plusieurs groupes
  - Un groupe est aussi un numéro
  - GID spécifiques
    - Privilèges pour accéder à des ressources spécifiques (audio, floppy, ...)

J'crois qu'j'ai mon compte

Hashage et Signature Numérique

- **Famille des algorithmes de chiffrement**
- **Appelés, aussi, algorithmes de chiffrement sans clé**
- **Transforme un message (chaîne de caractère, fichier) en une empreinte numérique de taille fixe (un grand entier de 128 bits ou plus)**

- **Transformation non réversible (fonction à sens unique)**
  - Impossible de retrouver le message d'origine

- **Problème des collisions**
  - Différents messages peuvent avoir une empreinte identique
  - S'il est possible de générer deux messages ayant la même empreinte, alors l'algorithme est « cassé »

- **Exemple : `md5sum`**
  - `md5sum /bin/bash` (fichier de 511 Ko) :
    `603492287ea2f26b9fb9266c961d5b0c  /bin/bash`

J'crois qu'j'ai mon compte

Fichiers de Gestion de Compte

- **`/etc/passwd` (lisible par tout le monde)**
  - *fred:x:1000:100:F. Combeau:/home/fred:/bin/bash*
  - Champ 1 : nom de login
  - Champ 2 : mot de passe hashé ou x si mot de passe dans shadow
    - * permet d'invalider un compte
  - Champ 3 : UID
  - Champ 4 : GID
  - Champ 5 : champs GECOS, informatif (suivant l'OS)
  - Champ 6 : répertoire d'accueil
  - Champ 7 : exécutable du shell de commande
    - Si présent dans `/etc/shells`, l'utilisateur peut le changer
    - Si shell spécifique, l'utilisateur ne peut pas le changer

J'crois qu'j'ai mon compte

- **`/etc/group` (lisible par tout le monde)**
  - *users:x:100:fred*
  - Champ 1 : nom du groupe
  - Champ 2 : mot de passe du groupe (x si mot de passe dans gshadow)
  - Champ 3 : GID
  - Champ 4 : liste de noms de login qui appartiennent au groupe

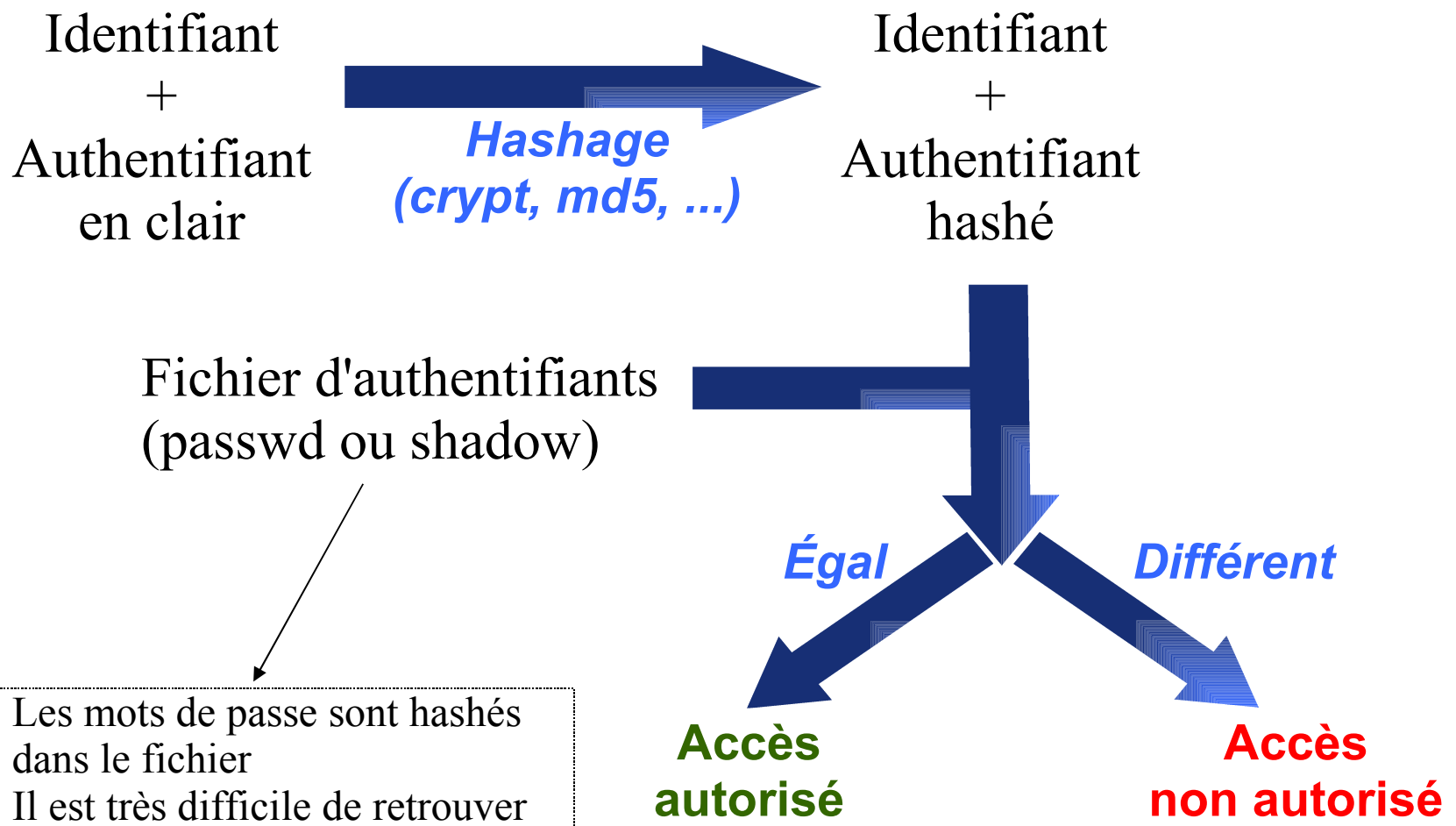
Fichiers de Gestion de Compte

- **`/etc/shadow` (lisible uniquement par le système)**
  - *fred:<password hashé>:12129:0:99999:7:::*
  - Champ 1 : nom de login
  - Champ 2 : mot de passe hashé (* : invalider un compte)
  - Champ 3 : dernier changement
  - Champ 4 : autorisation de changement de mot de passe
  - Champ 5 : mot de passe doit être changé
  - Champ 6 : avertissement utilisateur mot de passe à changer
  - Champ 7 : compte invalidé aprés expiration
  - Champ 8 : compte expiré
  - Champ 9 : réservé
    - Les champs 3 à 8 sont réservés à l'*expiration* des comptes
    - Commandes : `passwd` et `chage`

J'crois qu'j'ai mon compte

- **`/etc/gshadow` (lisible uniquement par le système)**
  - *users:*::*
  - Champs 1 : nom de groupe
  - Champs 2 : mot de passe hashé
  - Champs 3 : administrateurs du groupe
  - Champs 4 : membres du groupe

Synoptique d'Authentification



Les mots de passe sont hashés dans le fichier
Il est très difficile de retrouver le mot de passe en clair à partir de la version hashée.

J'crois qu'j'ai mon compte

Autres Fichiers Intervenant dans l'Authentification

- **`/etc/securetty`**
  - Points de connexion autorisés pour root (console, port série, tty...)
- **`/etc/shells`**
  - Shells autorisés (chemins complets)
- **`/etc/login.defs` (Linux)**
- **`/etc/login.conf` (BSD)**
  - Configuration des exécutables de la suite login
- **`/etc/nologin`**
  - La présence de ce fichier interdit la connexion des utilisateurs autres que root

J'crois qu'j'ai mon compte

PAM : Plugable Authentication Modules

- **Mécanisme d'authentification centralisé**
  - Les applications « PAMifiées » délèguent leur authentification
  - Les librairies PAM contrôlent cette authentification
  - La configuration des librairies PAM est effectuée et contrôlée par l'administrateur pour chaque application
  - Deux types de configuration :
    - `/etc/pam.conf` (fichier unique)
      - Plusieurs lignes par application
    - `/etc/pam.d` (dossier)
      - Un fichier de configuration par application utilisant les PAM

- **Intégration des PAM suivant la distribution**
  - Support des PAM
    - Redhat, Mandriva, Debian, FreeBSD, Solaris
  - Pas de support des PAM
    - Slackware, OpenBSD

J'crois qu'j'ai mon compte

- **4 catégories gérées**
  - **Authentification** : vérification de l'identité (couple identifiant/authentifiant)
  - **Compte** : vérification des informations de compte (mot de passe expiré, appartenance à un groupe)
  - **Mot de passe** : mis à jour du mot de passe (obliger l'utilisateur à changer de mot de passe après expiration)
  - **Session** : préparation de l'utilisation du compte (tracer la connexion, monter des répertoires utilisateurs)

- **4 contrôles de réussite**
  - **requisite** : tous ces modules DOIVENT réussir (en cas de non succès d'un module, le traitement de la pile s'arrête et renvoie le non succès)
  - **required** : Idem **requisite**, mais le traitement ira jusqu'à la fin de la pile
  - **sufficient** : la réussite de ce module suffit à valider la pile de modules
    - résultat ignoré si module “required” a échoué avant
  - **optional** : résultat considéré seulement si aucun autre module n'a réussi ou échoué

J'crois qu'j'ai mon compte

PAM

- **Modules PAM : `/lib/security`**
  - pam_unix.so, pam_nologin.so, pam_rootok.so...
- **Configuration des modules : `/etc/security`**
  - group.conf, limits.conf, pam_env.conf, time.conf...
- **Exemple : slackware – dropline**

```
% ls /etc/pam.d
dropline-installer  halt      poweroff  system-auth   xserver
gdm                 login     reboot    time-admin
gdm-autologin       other     samba     useradd
gdmsetup            passwd    shadow    xdm
gnome-system-log    pkgtool   su        xscreensaver
```

J'crois qu'j'ai mon compte

Exemple : login

```
auth        required     /lib/security/pam_securetty.so
auth        required     /lib/security/pam_stack.so service=system-auth
  auth        required     /lib/security/pam_env.so
  auth        sufficient   /lib/security/pam_unix.so likeauth nullok
  auth        required     /lib/security/pam_deny.so
auth        required     /lib/security/pam_nologin.so

account     required     /lib/security/pam_stack.so service=system-auth
  account     required     /lib/security/pam_unix.so

password    required     /lib/security/pam_stack.so service=system-auth
  password    required     /lib/security/pam_cracklib.so retry=3
  password    sufficient   /lib/security/pam_unix.so nullok md5 shadow use_authtok
  password    required     /lib/security/pam_deny.so

session     required     /lib/security/pam_stack.so service=system-auth
  session     required     /lib/security/pam_limits.so
  session     required     /lib/security/pam_unix.so
session     optional     /lib/security/pam_console.so
```

J'crois qu'j'ai mon compte

Exemples de Modules PAM

- **pam_unix**
  - Reproduit l'authentification Unix classique
  - auth
    - Compare le hash du mdp fourni avec passwd ou shadow
      - nullok : mdp vide accepté
      - nodelay : supprime le délai en cas d'échec
  - account
    - Vérifie le statut du compte de l'utilisateur dans shadow
  - password
    - Met à jour le mot de passe de l'utilisateur
      - md5 : utiliser le hash MD5 du mdp
      - shadow : utiliser le fichier shadow
      - nullok : permet de modifier le mdp à vide (sinon impossible)
  - session
    - Enregistre les évènements de connexion

J'crois qu'j'ai mon compte

Exemples de Modules PAM

- **pam_cracklib**
  - password
    - Vérifie la robustesse du mot de passe
      - retry : nombre d'essais en cas de mdp faible
      - difok : nombre minimal de caractères différents
      - minlen : taille minimale du mdp
- **pam_env**
  - auth
    - Configuration des variables d'environnement
- **pam_limits**
  - session
    - Configure les limites sur l'utilisation des ressources
- **pam_stack**
  - account, auth, password, session
    - Appelle la configuration PAM d'un autre service
      - service : service dont la configuration est utilisée

J'crois qu'j'ai mon compte

Les Mots de Passe à Usage Unique (OTP)

- **Le mécanisme OPIE (One-time Passwords In Everything) est inclus dans FreeBSD et OpenBSD**
  - Il est supporté par login, ftpd et su
  - Un module PAM le prend également en charge

- **L'algorithme S/KEY est utilisé pour générer les mots de passe à usage unique**
  - Lors de la phase de login, l'utilisateur reçoit un challenge
  - Il doit recopier ce challenge dans une calculatrice
  - Celle-ci fournit la réponse au challenge
  - L'utilisateur peut s'authentifier avec cette réponse

J'crois qu'j'ai mon compte

- **SU**
  - Permet de changer le compte sous lequel on est connecté (change UID et GID)
  - Configuration classique :
    - **`root`** peut prendre n'importe quelle identité sans mot de passe
    - Tout autre utilisateur peut prendre une autre identité en fournissant le bon mot de passe
  - La configuration de SU se fait par les PAM
    - Il est facile de changer la configuration par défaut
    - Exemple : restriction de SU à un groupe particulier
  - Lancé sans nom d'utilisateur, demande le passage sous `root`
  - Avec -, demande la création de l'environnement de l'utilisateur dont on demande l'identité
    - `% su -`
      - Passage sous le compte `root` avec l'environnement de `root`

J'crois qu'j'ai mon compte

- **SUDO**
  - Permet d'exécuter une commande sous l'identité d'un autre utilisateur
  - Permet en particulier l'utilisation de commandes spécifiques sous le compte `root`
  - Configuration par le fichier `/etc/sudoers`
    - utilisateur1 hôte = commande [utilisateur2]
    - Permet à l'utilisateur1 sur la machine hôte d'exécuter la commande commande en tant qu'utilisateur utilisateur2
    - Par défaut, utilisateur2 est `root`
  - Exemple : `fred ALL=/sbin/shutdown -h now`
  - Toute action effectuée à l'aide de la commande SUDO est enregistrée

J'crois qu'j'ai mon compte

Exploitation d'un système Unix

Gestion des fichiers

Permissions, ACL et Quotas

Permissions

- **A chaque fichier, sont associés un utilisateur propriétaire (UID) et un groupe propriétaire (GID)**
- **A chaque fichier, sont associés des permissions**
- **3 types de permissions :**
  - Lecture : `R`
  - Écriture : `W`
  - Exécution : `X`
- **`UMASK` permet de définir les droits hérités par défaut à la création des fichiers**

Permission denied

Permissions

- **3 groupes de droits qui s'appliquent :**
  - Au propriétaire du fichier
  - Au groupe propriétaire du fichier
  - Au reste du monde
  - `-rw-r-xr--  1 fred  users  51991 Mar 20 20:42 admin_unix.sxi`

- **Droits additionels :**
  - Set UID : s à la place de x pour le propriétaire
  - Set GID : s à la place de x pour le groupe propriétaire
  - Sticky Bit : t à la place de x pour le reste du monde
  - `-rwSr-sr--  1 fred  users  51991 Mar 20 20:42 libre_en_fete.sxi`

Permission denied

Permissions

- **Interprétation des permissions pour les fichiers**
  - Lecture : permet de lire le contenu du fichier
  - Écriture : permet d'écrire dans le fichier
  - Exécution : permet d'exécuter le fichier sous son identifiant
  - Set UID : permet d'exécuter le fichier sous l'identifiant du propriétaire du fichier
  - Set GID : permet d'exécuter le fichier sous l'identifiant de groupe du groupe propriétaire du fichier
  - SUID et SGID sont à éviter car peuvent engendrer des problèmes de sécurité

Permission denied

Permissions

- **Interprétation des permissions pour les dossiers**
  - Lecture : permet de lire le nom des fichiers composant un répertoire
  - Écriture : permet de créer et d'effacer des fichiers dans un répertoire
  - Exécution : permet d'accéder aux informations des fichiers composant un répertoire et de s'y arrêter
  - Set GID : permet de créer des fichiers dont le groupe propriétaire est celui du répertoire
  - Sticky Bit : permet d'effacer uniquement les fichiers dont on est propriétaire

Permission denied

- **Les commandes importantes**
  - `chown` : change le propriétaire d'un fichier (ainsi que le groupe)
    - `chown toto /home/toto`
    - `chown toto:users fichier`
  - `chgrp` : change le groupe du fichier
  - `chmod` : modifie les permissions sur un fichier
    - deux façons de procéder
      - spécifier les permissions
        `chmod g+rx,o-rwx fichier`
      - utiliser le masque
        `chmod 755 fichier`

Permission denied

Attributs

- **Attributs de fichiers**
  - dépendent du système de fichiers
  - meta-info influençant le comportement du système
  - ext2/3 : `lsattr/chattr`

- **Détails des attributs (Linux) [ASacDdlijsTtu]**
  - `A`
    - Access Time pas mis à jour (gain de performance)
  - `a` *(root only)*
    - Append only : fichier ouvrable seulement en APPEND
  - `i` *(root only)*
    - Immutable : fichier ne peut être modifié, supprimé, hard-linké, ...
  - `S`
    - Sync : comme le sync du mount(), mais pour des fichiers
  - `u`
    - Undelete : sauvegarde du fichier pour récupération

Permission denied

ACLs Étendues

- **Les ACL permettent d'étendre le modèle des permissions**
  - Il est possible de définir des listes d'accès, accordant des droits particuliers à certains groupes ou utilisateurs
  - Toutefois, cela complique l'administration
  - Tout utilisateur peut définir des ACL sur ses fichiers
  - Disponible sous Linux (noyau à recompiler)
  - Commandes:
    - `getfacl`
    - `setfacl`

- **Exemple**

```
%setfacl -m u:titi:rwx fichier
%ls -l fichier
-r-xr-xr-x+   1 toto users   216112 Jan 27 09:54 fichier
%getfacl fichier
user::r-x
user:titi:rwx
group::r-x
mask:rwx
other:r-x
```

Permission denied

Quotas

- **Les quotas permettent de limiter l'utilisation des systèmes de fichier par les utilisateurs**
  - Limites sur inodes et sur blocs
  - Deux types de limites :
    - Soft limit
      - Limite «souple» : permet d'accorder une période (grace period) pendant laquelle l'utilisateur peut dépasser la limite
    - Hard limit
      - Limite «dure» : l'utilisateur ne peut dépasser cette limite, le système refusera d'accorder les inodes ou blocs
  - Quotas par utilisateur ou par groupe
  - Les systèmes de fichier doivent être montés avec l'option quota (ou plus spécifiques usrquota et grpquota)
  - Les quotas doivent être gérés par le système de fichier
    - ext(2|3) et reiserfs sous Linux

Permission denied

- **Commandes**
  - `quotacheck` : met à jour les quotas, à utiliser au démarrage du système, puis périodiquement (toutes les semaines)
  - `quotaon` : active les quotas pour un système de fichier donné
    - `quotaon -a` (active les quotas suivant fstab)
  - `quotaoff` : désactive les quotas
  - `edquota` : permet de définir les quotas par utilisateur (lance l'éditeur sur le fichier de quota)
    - edquota -u toto (définition du quota de l'utilisateur toto)
    - edquota -g users (quota pour le groupe users)
    - edquota -t (configuration de grace period)
  - `repquota` : rapport sur les quotas

Permission denied

Exploitation d'un système Unix

Planification de Tâches

CRON, AT et Scripts d'Exploitation

CRON

- **cron est un démon permettant la programmation de tâches exécutées périodiquement**
  - cron est lancé au démarrage du système
  - cron peut être configuré pour lancer des travaux chaque jour, chaque semaine, chaque mois
  - La configuration est faite par des fichiers `crontab`
  - Configuration globale : `/etc/crontab`
    - Parfois la crontab de root est utilisée
  - Chaque utilisateur peut définir des tâches périodiques
    - Emplacement des fichiers crontab :
      - Sous Linux : `/var/spool/cron/<user>`
      - Sous BSD : `/var/cron/tabs/<user>`
  - L'utilisateur utilisé pour lancer les commandes est le propriétaire du fichier, sauf pour la crontab globale

Exécution décalée

- **Structure du fichier crontab :**
  - Commentaires avec #
  - Liste de variables d'environnement
    - SHELL=<commande shell à utiliser>
    - HOME=<dossier d'exécution par défaut>
    - MAILTO=<adresse d'envoi du rapport>
  - Lignes de tâches
    - *<minutes> <heures> <jours-du-mois> <mois> <jours-de-la-semaine> <tâche>*
  - Dans `/etc/crontab`, l'utilisateur utilisé pour lancer la tâche est spécifié avant la tâche elle-même

- **Exemples de tâches programmées :**
  - `* * * * 1 quotacheck` (quotacheck tous les lundis)
  - `* * 1 * * updatedb` (mise à jour de la base locate le 1° du mois)

  - à noter : il existe des alias @daily, @weekly, @monthly, @yearly

- **Commandes**
  - `crond`
    - C'est le démon lui-même, il est lancé par l'un des scripts de démarrage, sans option
  - `crontab`
    - C'est le programme de manipulation des crontab, qui modifie la crontab de l'utilisateur courant
    - `crontab -l` : affiche la crontab
    - `crontab -r` : efface la crontab
    - `crontab -e` : lance l'éditeur sur la crontab et valide le contenu en sortie

AT

- **at permet de lancer ponctuellement des commandes, à une date et une heure données**
  - Le démon `atd` est lancé par les scripts de démarrage
  - La commande `at` permet de programmer le lancement d'une tâche (une seule fois, pas de périodicité)
    - Utilisation : `at HEURE`
      `at midnight, at noon, at 4pm`
      `at + 5 minutes, at + 2 days`
    - puis saisir la liste des commandes à effectuer
    - `atq` affiche la liste des tâches programmées
    - `at -c <numéro de tâche>` affiche le contenu d'une tâche

Scripts d'Exploitation

- **L'ensemble des tâches d'exploitation du système peut être effectué par des scripts**
  - Les scripts peuvent être programmés dans tout langage
    - Par exemple : Shell, Perl, Python, ...
  - Le lancement des scripts peut être automatisé par `cron` et `at`.
  - Exemples
    - Vérification des comptes sans mot de passe
    - Utilisation du disque
    - Sauvegardes du système
  - Les scripts doivent être testés le plus possible avant d'être déployés
    - Test sur des copies des vrais fichiers, dans une arborescence à part, tester les limites, ...
  - Les scripts ne peuvent pas tout faire, il faut parfois revenir à un langage de programmation (bien que le cas se présente rarement)

Exécution décalée

Exploitation d'un système Unix

Le Noyau

Fonctionnement, Modules, Configuration et Compilation

- **Le noyau est le coeur du système d'exploitation**
  - Il est chargé au démarrage de la machine (cf. Boot)
  - Il fournit les interfaces de communication avec les périphériques matériels
  - Il peut être monolithique (un seul bloc) ou modulaire
  - Emplacement
    - Sous Linux : `/boot/vmlinuz` (compressé)
    - Sous BSD : `/boot/kernel/kernel` (non compressé)
  - Chargement
    - Lilo

```
image = /boot/vmlinuz
  root = /dev/hda7
  label = Linux
  read-only
  vga = 834
  initrd = /boot/initrd
```

    - Grub

```
title Linux
  kernel (hd0,6)/boot/vmlinuz root=/dev/hda7 ro vga=834
  initrd (hd0,6)/boot/initrd
```

Everything about kernel

- **Linux : le boot loader (lilo, grub) peut passer des paramètres de démarrage au noyau (*ligne de commande du noyau*)**
  - `single` : démarrage en mode single user (runlevel 1)
  - `emergency` : un shell est lancé à la place d'init
  - `ro` : système de fichier racine monté en lecture seule (recommandé)
  - `vga=xxx` : choix du mode graphique (`normal` pour le mode texte)
  - `initrd` : chargement d'un système de fichier en mémoire avant le lancement du noyau

Everything about kernel

- **La commande sysctl permet de modifier les paramètres du noyau pendant le fonctionnement du système**
  - Initialisation selon le fichier `/etc/sysctl.conf`
  - `sysctl <variable>` affiche la valeur d'une variable
  - `sysctl -a` pour voir toutes les variables (combiner avec `grep`)
  - Pour modifier une valeur :
    - `sysctl -w variable=valeur` sous Linux
    - `sysctl variable=valeur` sous *BSD

- **`/proc` contient des informations sur les processus en cours d'exécution**
  - Particularité de Linux : les paramètres du système peuvent être configurés par des entrées dans /proc
  - Exemples
    - `/proc/sys/net`, `/proc/modules`, `/proc/meminfo`

Everything about kernel

- **Si le noyau n'a pas été compilé de façon monolithique, il est possible de charger des modules pour étendre les fonctionnalités du noyau après le démarrage du système**
  - Exemple : les pilotes de périphériques
  - Commandes :
    - Sous Linux
      - `lsmod` (affiche la liste des modules chargés)
      - `insmod` (charge un module)
      - `modprobe` (charge un module avec les modules pré-requis)
      - `rmmod` (retire un module à condition qu'il ne soit plus utilisé)
      - `/lib/modules/<version du noyau>` (contient les modules)
    - Sous *BSD
      - `kldstat`, `kldload`, `kldunload`
      - `/boot/kernel` contient les modules
  - Chargement automatique possible

Everything about kernel

- **Le code source du noyau de Solaris n'est pas fournit, cependant des modules peuvent être insérés**
  - Il est impossible de recompiler le noyau Solaris
  - Ce noyau fournit des interfaces pour le chargement de modules, il est donc possible de charger de nouveaux pilotes
  - Les modules ne sont chargés que lors du démarrage, il faut donc spécifier dans le fichier de configuration `/etc/system` le nouveau module à charger puis relancer la machine
  - Le noyau et les modules sont dans `/kernel`, le noyau est `/kernel/unix`

Everything about kernel

Utiliser un Noyau Modifié

- **Requis dans certains cas**
  - Besoin d'une fonctionnalité d'une nouvelle version
  - Application d'un patch
  - Correction d'un problème de sécurité
  - Pas de mise à jour constructeur
  - Optimisation pour une architecture
- **Précautions à prendre**
  - Conserver un ancien noyau fonctionnel
    - pas toujours possible
    - sauvegarder la configuration de l'ancien noyau
  - Sauvegarde éventuelle des partitions
    - en cas de corruption du système de fichiers
- **Opération longue**
  - Nombre d'options activées
  - Nombre de processeurs (`make -j N`)
- **Dossier « `Documentation` »**

Everything about kernel

- **Avant la configuration**
  - Enumération des périphériques matériels
    - `lspci`
    - `dmesg`
  - Choix des systèmes de fichiers
  - Déterminer les protocoles réseau nécessaires
  - Installer les sources du noyau
  - Appliquer les patches

- **Configuration**
  - Copier le fichier de configuration choisi en `.config`
  - `make menuconfig`
  - Modulaire / Monolithique
  - Important
    - Pilotes pour le disque dur et le système de fichier racine
      - ou utiliser un initrd
    - `CONFIG_HIGHMEM4G` si 1Go de RAM ou plus

Everything about kernel

- **Avant la compilation**
  - Vérifier l'espace libre : ~ 200 Mo nécessaires
  - Éditer `Makefile` pour certaines options spécifiques
    - Suffixe de version du noyau

- **Compilation**
  - Noyau 2.6.xx : `make`
    - `bzImage modules`
      - compilation du noyau compressé
      - compilation des différents modules sélectionnés
    - `make O=<dir>` : fichiers produits dans *dir*
  - Noyau 2.4.xx
    - `make bzImage`
    - `make modules`
    - Très verbeux

Installation d'un Noyau

- **Avant l'installation**
  - Monter éventuellement la partition `/boot`
  - Vérifier si `lilo` ou `grub` est utilisé

- **Installation**
  - Automatique : `make install`
    - exécute `/sbin/installkernel`
    - Comportement suivant la distribution (peut être dangereux)
  - A la main
    - copier le noyau : `arch/<arch>/boot/bzImage`
      - `/boot/vmlinuz-<version>`
    - copier `System.map`
      - `/boot/System.map-<version>`
    - copier `.config`
      - `/boot/config-<version>`

Everything about kernel

Tester le Nouveau Noyau

- **Mettre à jour le bootloader**
  - Ajouter le nouveau noyau
  - Préserver les anciennes entrées
  - Relancer l'installation du bootloader (s'il s'agit de `lilo`)
- **Redémarrer la machine**
  - Sélectionner le nouveau noyau
  - Diagnostiquer les erreurs ...

Exploitation d'un système Unix

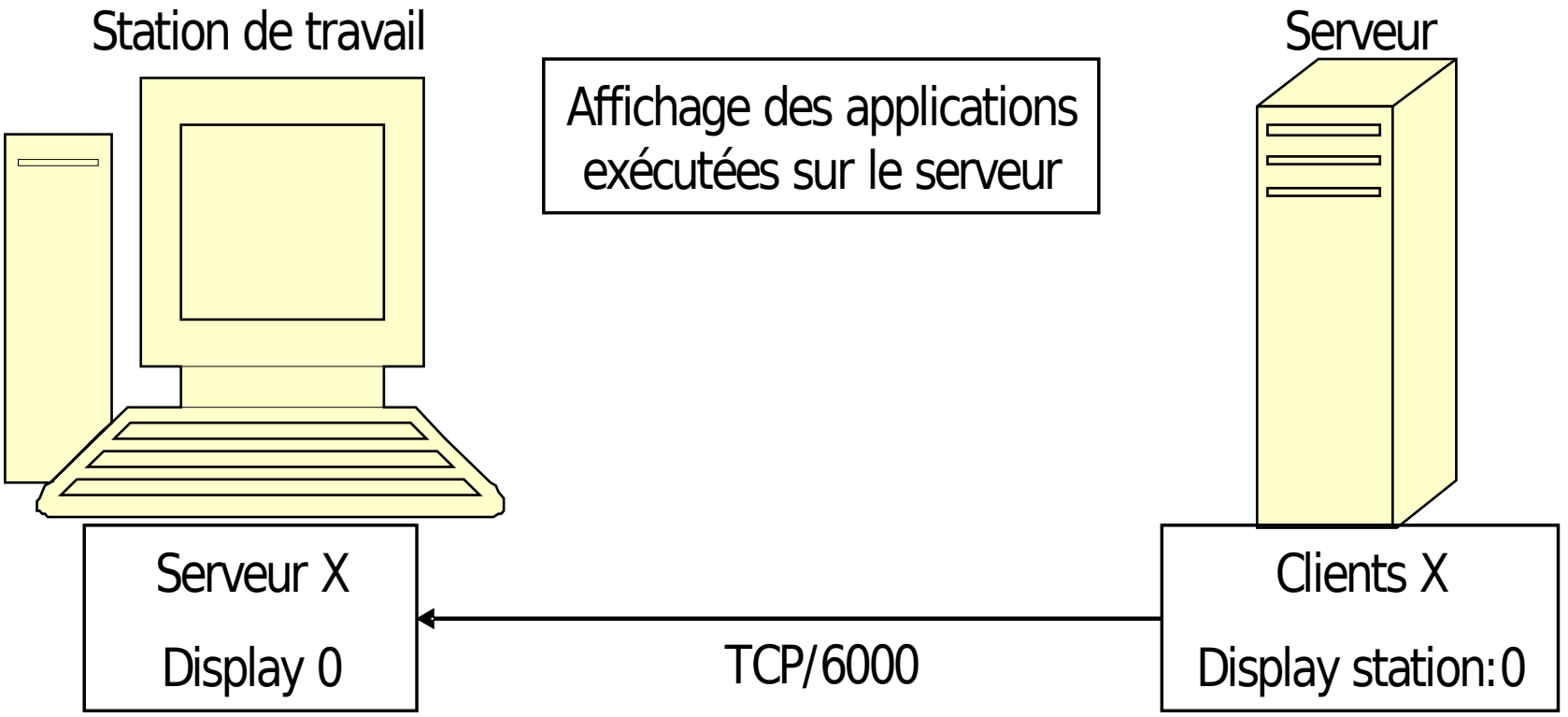
X-Window System

Architecture, Modules et Configuration

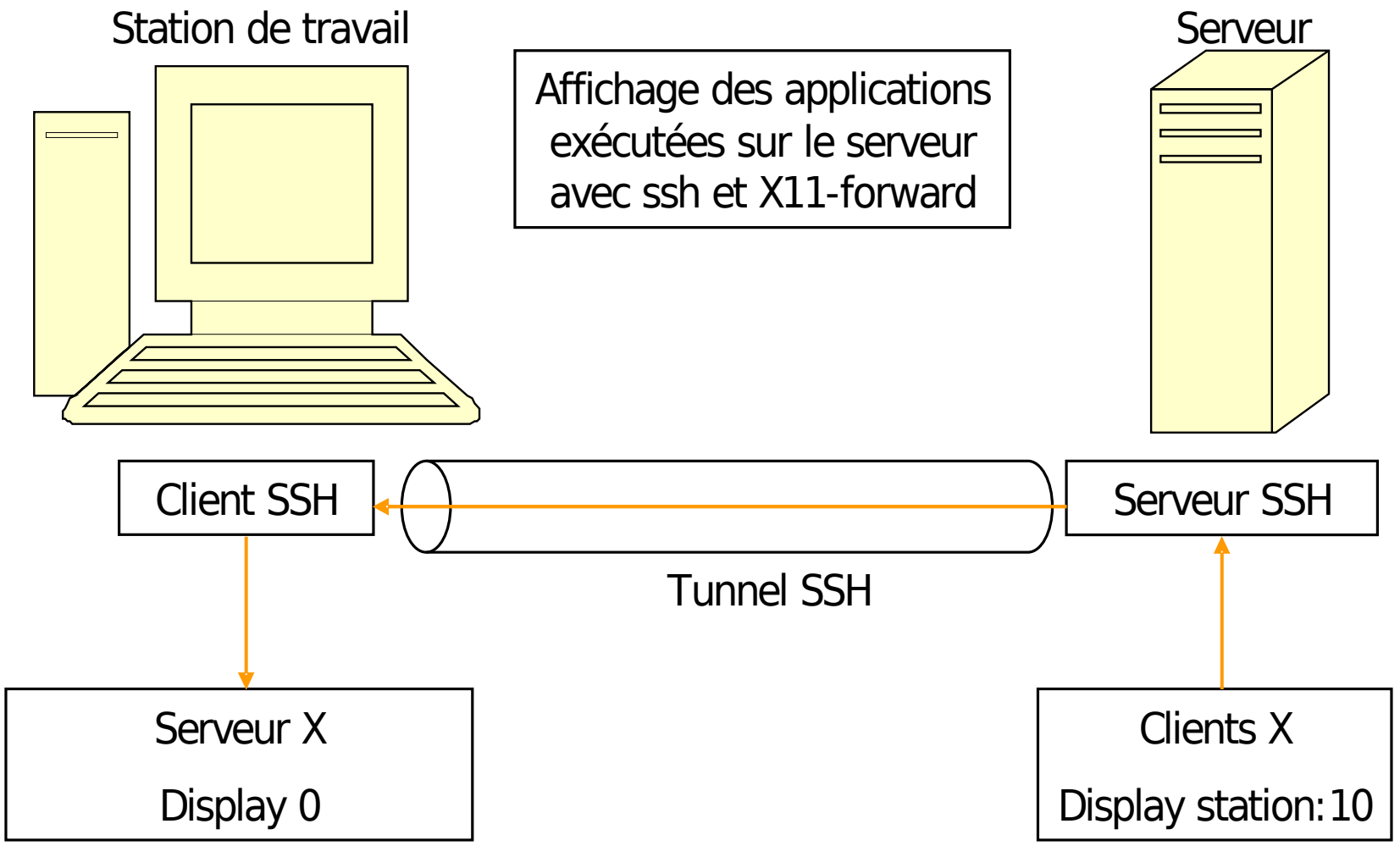
Architecture de X-Window

- **X11 est X Window System version 11**
- **X-Window est une architecture client-serveur, destinée à la gestion des environnements graphiques**
  - Le serveur X effectue le rendu graphique sur une machine donnée, en utilisant le matériel (carte graphique, écran, clavier, souris...)
  - Les clients X (gestionnaires de fenêtres, terminaux X...) se connectent au serveur X pour leur affichage
  - La communication se fait localement par un socket UNIX, mais peut aussi être faite par TCP/IP pour un serveur distant (ports TCP 6000 et suivants)
  - La variable d'environnement qui désigne le serveur X est `DISPLAY` (sur une machine autonome, `DISPLAY=:0`)
- **Implémentations libres actuelles (toujours X11R6)**
  - XFree86 (version 4.4)
  - X.org (version 7.4)

Architecture Client/Serveur



Architecture Client/Serveur + Tunnel SSH



Contrôle d'Accès des Clients

- **Si l'accès est autorisé, un client à un accès TOTAL au serveur X (et aux autres clients connectés à ce serveur)**
  - Politique du TOUT ou RIEN
  - Problème potentiel de sécurité

- **Contrôle par adresse IP**
  - La commande `xhost` permet d'accorder l'accès par adresse IP
    - `xhost +` : toutes les machines ont accès
    - `xhost -` : aucune machine n'a accès
    - `xhost + toto` : ajout de la machine toto en accès

- **Contrôle par un secret partagé**
  - L'utilisateur obtient un cookie auprès du serveur (généré au lancement de celui-ci), qui lui permet de s'y connecter
  - Les cookies sont stockés dans `~/.Xauthority`
  - La commande xauth permet de gérer les cookies

Modules d'un Serveur X

- **Le serveur X a une architecture modulaire**
  - Les différents pilotes utilisés par le serveur X sont des modules
  - Différents types de modules
    - Cartes graphiques
    - Périphériques d'entrée (clavier, souris, ...)
    - Extensions (freetype, GLX, ...)
  - Situés dans `/usr/X11R6/lib/modules/`
  - Les modules sont chargés au démarrage du serveur X suivant la configuration
  - L'échec du chargement d'un module n'est pas forcément fatal, mais entraîne le non-fonctionnement du périphérique associé
    - Fatal s'il s'agit d'un module de carte graphique

- **La configuration du serveur X se fait par un fichier**
  - **Pour Xorg**
    - `/etc/X11/xorg.conf`
  - **Pour XFree**
    - `/etc/X11/XF86Config[-4]`

- **Le fichier se compose de sections**
  - **Section Files**
    - Définition des ressources rgb, polices
  - **Section Module**
    - Extensions à charger (GLX, extmod)
  - **Sections InputDevice**
    - Définition des périphériques d'entrée et de leurs pilotes (souris, clavier, autres)
  - **Sections Monitor**
    - Définition des écrans et de leurs modes de fonctionnement

- **Le fichier se compose de sections**
  - **Sections Device**
    - Définition des cartes graphiques et pilotes
  - **Sections Screen**
    - Associe cartes et écrans à des résolutions de fonctionnement
  - **Sections ServerLayout**
    - Associe une section **Screen** à des périphériques d'entrée, constitue le point d'entrée pour le démarrage du serveur X
      - On peut spécifier le layout sur la ligne de commande
        - `X -layout <layout>`

Exploitation d'un système Unix

Sauvegarde et Restauration

Politiques et Commandes

Politiques de Sauvegarde

- **Pourquoi sauvegarder?**
  - Parce que le matériel tombera en panne (disque dur défectueux, panne générale d'un système)
  - Parce que les utilisateurs ne sont pas infaillibles (ils effacent des fichiers par erreur)

- **Quand sauvegarder?**
  - Il faut adapter la politique de sauvegarde à l'importance des données (station de travail, serveur de calcul, base de données)
  - Il faut autant que possible sauvegarder pendant les heures creuses (nuit, week-end)
  - Bien utiliser les différents types de sauvegardes
  - Aucune politique n'est applicable partout, il faut toujours s'adapter à la situation

- **Fil conducteur**
  - **Envisager la perte de tous les disques durs pendant que vous n'êtes pas connecté sur le système**

Il faut sauvegarder

Politiques de Sauvegarde

- **Les différents types de sauvegardes**
  - **Sauvegarde totale**
    - Tous les fichiers sont sauvegardés
    - Pour une restauration, il suffira de tout redescendre
    - Sauvegarde longue et nécessitant beaucoup de ressources (bandes)
    - Restauration facile et rapide
  - **Sauvegarde incrémentale**
    - Seuls les fichiers qui ont été modifiés depuis la dernière sauvegarde totale, sont sauvegardés
    - Pour une restauration, il faut faire une restauration totale + une restauration incrémentale, à partir de la restauration totale
    - Sauvegarde courte et nécessitant peu de ressources
    - Restauration complexe et pouvant être longue

Il faut sauvegarder

- **Les différents types de sauvegardes**
  - **Sauvegarde différentielle**
    - Seuls les fichiers qui ont été modifiés depuis la dernière sauvegarde incrémentale, sont sauvegardés
    - Pour une restauration, il faut faire une restauration totale + une restauration incrémentale + une restauration différentielle
    - Sauvegarde très courte et nécessitant très peu de ressources
    - Restauration très complexe et pouvant être très longue

Il faut sauvegarder

- **Les différents types de sauvegardes**
  - **Sauvegarde système**
    - Consiste à sauvegarder un système complet dont l'OS
    - Nécessite le plus souvent de passer en single user
  - **Sauvegarde de données**
    - Les homes utilisateurs, des applications tierces, archivage
  - **Sauvegarde applicative**
    - SGBD, serveur Web
    - Nécessite une procédure spécifique pour sauvegarder au niveau de l'application, sinon l'application devra être arrêtée avec un script adaptée

Il faut sauvegarder

Politiques de Sauvegarde

- **Questions importantes**
  - Quels fichiers doivent être sauvegardés ?
  - Quelle est la taille d'une sauvegarde totale, incrémentale ou différentielle ?
  - Où sont ces fichiers ?
  - Qui sauvegarde les fichiers ?
  - Où sont faites les sauvegardes ?
  - Quelle est la fréquence de modification des fichiers à sauvegarder?
  - Quel est le délai de restauration à tenir ?
  - Où les sauvegardes sont-elles stockées ?
  - Combien me faut-il de bandes (DVD, CD-RW, ...) ?
  - Combien de temps dure une sauvegarde totale ?

- **Stratégies de haute disponibilité**
  - La restauration du système doit respecter des délais très courts en cas de panne

Il faut sauvegarder

Politiques de Sauvegarde - Exemple

- **Sauvegarder un serveur NFS exportant les homes utilisateurs**
  - Configuration
    - Linux Redhat 5 U4, installé avec KickStart
    - 1 lecteur de bandes DLT 700 Go
    - 1 DD de 136 Go pour le système
    - 3 DD de 300 Go en RAID-5 matériel pour le `/export/home`
  - **Sauvegarde des homes utilisateurs**
    - Sauvegarde totale le 1er WE du mois
      - 1 bande/mois
    - Sauvegarde incrémentale les 3 (ou 4) autres WE du mois
      - 3 bandes/mois
    - Sauvegarde différentielle tous les soirs de la semaine
      - 4 bandes/semaine soit 16 bandes/mois
    - TOTAL ⇨ 16+3+1 = 20 bandes/mois

Il faut sauvegarder

Politiques de Sauvegarde - Exemple

- **Sauvegarder un serveur NFS exportant les homes utilisateurs**
  - Configuration
    - Linux Redhat 5 U4, installé avec KickStart
    - 1 lecteur de bandes DLT 700 Go
    - 1 DD de 136 Go pour le système
    - 3 DD de 300 Go en RAID-5 matériel pour le `/export/home`
  - **Sauvegarde système**
    - L'installation par KickStart permet une restauration plus rapide du système
    - Sauvegarde des fichiers spécifiques au serveur (`/etc`) et du fichier KickStart d'installation
    - En cas de crash du disque dur système
      - Ré-installation du serveur via KickStart
      - Récupération des fichiers spécifiques

Il faut sauvegarder

DUMP

- **dump est un outil permettant de sauvegarder des systèmes de fichier entier**
  - Le programme dump doit être adapté au système de fichier à sauvegarder
    - Exemple : dump sous Linux ne sauvegarde que ext2/3
  - Il permet des sauvegardes incrémentales
    - `dump -0` : garantit une sauvegarde complète
    - `dump -1` : sauvegarde les fichiers modifiés depuis la dernière sauvegarde de niveau 0
    - Attention : le niveau par défaut est 9
  - Le fichier `/etc/dumpdates` contient les dates et niveaux des dernières sauvegardes
    - `dump -W` : informations sur les sauvegardes à faire
    - `dump -u` : met à jour `dumpdates` si la sauvegarde réussit
  - Dump sous Linux peut également sauvegarder des fichiers ou dossiers particuliers, mais seulement de niveau 0 et sans mettre d'entrée dans `dumpdates` (préférer tar/cpio)

Il faut sauvegarder

RESTORE

- **`restore` est le pendant de dump, il permet la restauration des fichiers sauvegardés avec dump**
  - `-C` : compare les fichiers sauvegardés et les fichiers actuels
  - `-r` : restaure le système de fichier entier (formaté au préalable)
  - `-i` : restauration interactive
  - `-x` : restauration de certains fichiers (préférer tar ou cpio)
  - `-f` : spécifie le fichier

TAR

- **`tar` (tape archiver) est un outil de création d'archives adapté à la sauvegarde d'arborescences de fichiers**
  - A l'origine, tar est fait pour copier des arborescences de fichier sur un lecteur de bande
  - Sous Linux et *BSD, tar est aujourd'hui un utilitaire d'archivage très souple, avec possibilité de compression
  - options:
    - c : création d'archive
    - x : extraction d'archive
    - f : spécification du fichier
    - v : mode verbeux
    - z : compression gzip
    - j : compression bzip2 (plus lent mais plus efficace)
  - Anecdotique :
    - tar c : écriture type BSD
    - tar -c : écriture type GNU

Il faut sauvegarder

- **Exemples**
  - `tar c /home`
    - Simple sauvegarde de /home sur le périphérique de bande par défaut (`/dev/rmt0`)
  - `tar czf projet.tar.gz projet`
    - Sauvegarde le dossier projet dans le fichier projet.tar.gz (compressé au format gzip)
  - `tar xvjf linux-2.6.24.tar.bz2 -C /usr/src`
    - Extraction des fichiers de l'archive linux-2.6.24 (compressée bzip2) dans le dossier /usr/src, en mode verbeux (les fichiers extraits sont écrits au fur et à mesure sur le terminal)
  - `tar xj -p -f sauvegarde.tar.bz2`
    - Extraction de l'archive sauvegarde avec restauration des permissions
  - `tar x <fichier à restaurer>`
    - Restauration d'un fichier précis à partir de la bande

CPIO

- **`cpio` est comparable à tar, mais il est conçu pour sauvegarder des listes de fichiers plutôt que des arborescences**
  - `cpio` est habituellement combiné avec `find`
    - Exemple : `find /chemin | cpio`
  - `cpio` peut utiliser de nombreux formats d'archive, dont `tar`
  - `cpio` ne gère pas les formats compressés, mais il peut être combiné avec `zcat` ou `bzcat`
  - Options :
    - `cpio -o` : création d'archive
    - `cpio -i` : restauration
    - `-F` : spécification de l'archive
    - `-H` : format d'archive (tar par défaut)

Il faut sauvegarder

DD

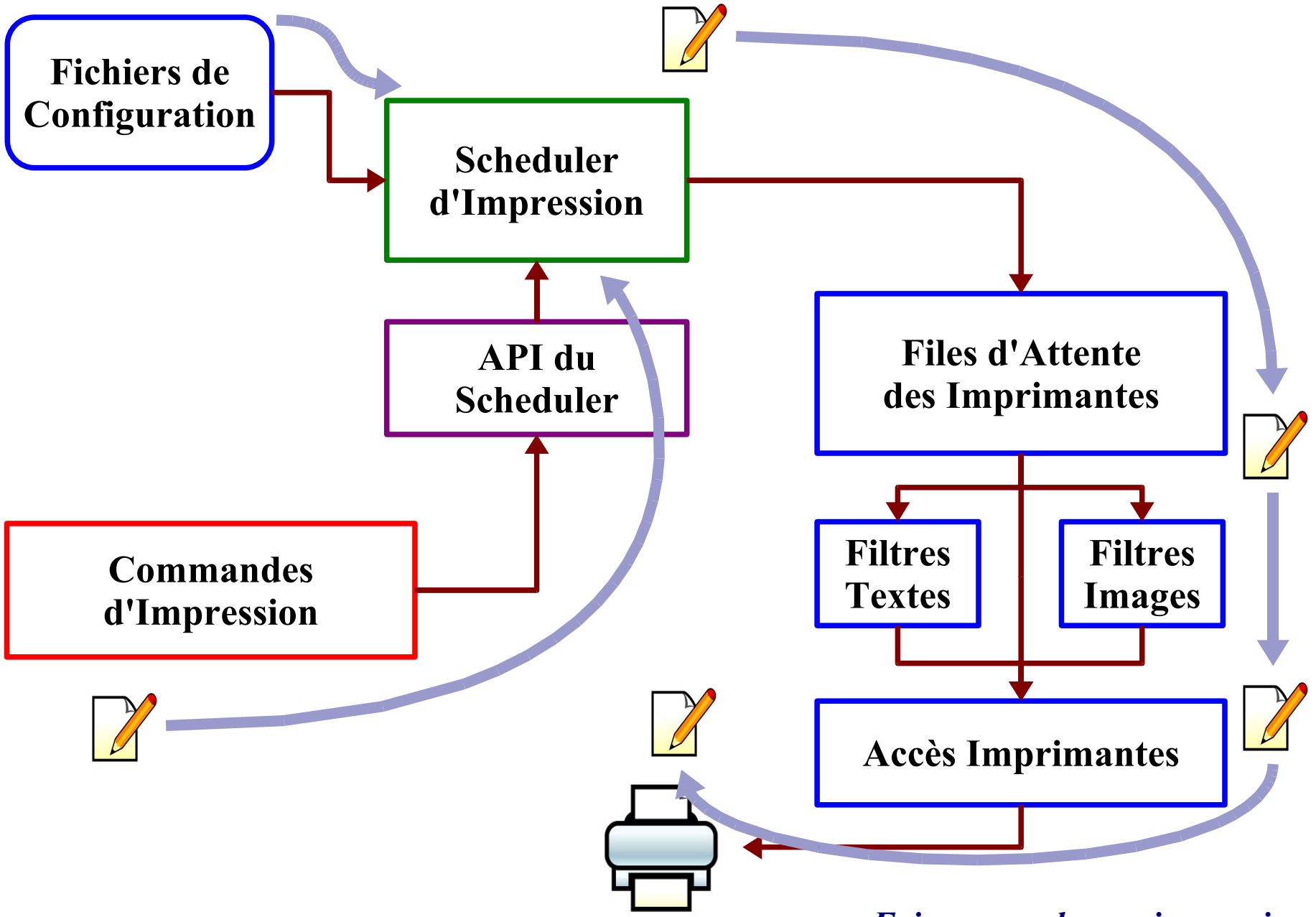
- **`dd` est un outil de copie de bas niveau, il copie des blocs de données**
- **Abréviation de Disc Dump**
  - On utilise dd pour faire des copie bas niveau de système de fichiers
    - `dd if=/dev/hda of=fichier_de_sauvegarde`
    - Cette commande extrait l'image brute du disque hda (bloc par bloc) vers un fichier de sauvegarde

Exploitation d'un système Unix

Gestion des Impressions

LPD et CUPS

Principes Généraux d'un Serveur d'Impression



Faisons une bonne impression

- **lpd est le système d'impression *BSD**
  - Il repose sur le démon d'impression `lpd`, et le fichier de configuration `/etc/printcap`
  - Le fichier `/etc/printcap` définit pour chaque imprimante comment on y accède (port parallèle, réseau...), les filtres à appliquer pour les travaux d'impression
  - Le démon `lpd` est responsable du *spooling* et de l'envoi des travaux à l'imprimante
  - Les commandes accessibles aux utilisateurs :
    - `lpr` : envoie un travail d'impression au démon
    - `lpq` : affiche la file d'attente des travaux d'impression
    - `lprm` : permet d'annuler un travail

- **Cups hérite du système d'impression de System V**
  - cups définit des classes d'imprimantes
  - Ensuite, on peut attacher des imprimantes à des classes, avec un emplacement précis (port parallèle, réseau...)
  - Les fichiers de configuration sont dans `/etc/cups`
    - Le fichier `printers.conf` contient la définition des imprimantes avec leur emplacement
    - Le fichier `classes.conf` contient la définition des classes avec les imprimantes qui appartiennent à ces classes
  - Commandes utilisateur :
    - `lp` : envoie un travail d'impression
    - `lpstat` : informations sur les files d'attente des imprimantes
    - `cancel` : annule un travail d'impression

Faisons une bonne impression

Exploitation d'un système Unix

Configuration du Réseau

- **Configuration des interfaces TCP/IP**
  - `ifconfig <interface> <adresse IP> netmask <masque réseau>`
  - La passerelle par défaut est configurée par les commande de routage
    - Linux : `route add default gw <adresse IP>`
  - La station peut être cliente DHCP
    - Une requête DHCP devra être envoyée par la station pour récupérer sa configuration réseau (commande `dhclient`) via un serveur DHCP
- **Listing des services réseau**
  - `netstat`
    - `-a` : liste les sockets non connectés (en écoute)
    - `-t` : TCP
    - `-u` : UDP
    - `-p` : donne le PID associé au socket

- **La configuration DNS est importante pour le bon fonctionnement d'un système UNIX (peut être fourni par le serveur DHCP)**
  - Fichier `/etc/host.conf`
    - définit l'ordre de recherche des noms DNS
    - `order hosts, bind`
      - signifie que l'on regarde d'abord dans le fichier /etc/hosts puis ensuite on interroge le serveur DNS
  - Fichier `/etc/nsswitch.conf`
    - Peut, également, définir l'ordre de recherches des noms DNS
      - `hosts: files dns`
  - Fichier `/etc/resolv.conf`
    - Spécifie l'adresse IP des serveurs DNS
      - `nameserver <adresse IP>`
    - Spécifie les domaines de recherche de noms DNS
      - `search <domaine>`

Exploitation d'un système Unix

Les Journaux d'Evènements Système

Syslog, Logrotate et Accounting

SYSLOG

- **`syslog` reçoit l'ensemble des logs du système et les répartit dans différents fichiers suivant sa configuration**
  - Le démon `syslogd` est lancé au démarrage
  - Deux façons de recevoir les logs :
    - `/dev/log` : socket unix
    - Socket UDP port 514 : pour le réseau
  - Chaque message possède une *facility* et une *priority* :
    - facility indique le type de log (kernel, daemon, ...)
    - priority indique la gravité du message (debug, emerg, ...)
  - Le fichier `/etc/syslog.conf` définit la configuration de syslog
    - Par facility et/ou par priority
    - Possibilité de diriger les logs dans des fichiers ou de les transmettre à d'autres programmes

I'm watching you

LOGROTATE

- **`Logrotate` contrôle la taille des fichiers de log et les sauvegardes au besoin**
  - Lancement périodique par `cron`
    - Dans la crontab de root

```
40 4 * * * /usr/bin/run-parts /etc/cron.daily 1> /dev/null
```

    - Dans le fichier /etc/cron.daily/logrotate

```
#!/bin/sh
/usr/sbin/logrotate /etc/logrotate.conf
```

  - Configuration de logrotate : `/etc/logrotate.conf`

Accounting

- **Les système de comptabilité (ou *accounting*) permettent de surveiller l'utilisation des ressources par les utilisateurs à des fins de statistiques ou de facturation (parfois le temps CPU n'est pas donné)**
  - La comptabilité classique stocke ses informations dans
    - `/var/adm` (pour les *BSD)
    - `/var/log/*acct` (pour Linux)
  - Les commandes permettant l'extraction des informations :
    - `accton` : activer ou désactiver l'accounting
    - `sa` : informations sur l'utilisation CPU et mémoire
    - `ac` : temps de connexion des utilisateurs
    - `lastcomm` : dernières commandes lancées
    - `acctail` : surveillance interactive de l'accounting
  - Linux : options `CONFIG_BSD_PROCESS_ACCT` et `CONFIG_BSD_PROCESS_ACCT_V3`

I'm watching you

BIBLIOGRAPHIE

Ouf, c'est enfin terminé

Bibliographie

- **Les deux Bibles :**
  - Les bases de l'administration système, 3ème Edition
    - Æleen Frich – O'Reilly – 2841772225
  - Unix System Administration Handbook, 3rd Edition
    - Evi Nemeth, Scott Seebass, Garth Snynder
    - Prentice Hall PTR – 0130206016

- **Le système Linux :**
  - Le système Linux, 4ème Edition
    - Matt Welsh, Matthias Kalle Dalheimer, Terry Dawson et Lar Kaufman
    - O'Reilly – 2-84177-241-1
  - Administration réseau sous Linux, 2ème Edition
    - Olaf Kirch et Terry Dawson – O'Reilly – 2-84177-125-3
  - Linux Administration Handbook
    - Evi Nemeth, Garth Snyder, Adam Boggs
    - Prentice Hall – 0130084662

J'aime lire

Bibliographie

- **Les BSDs :**
  - Absolute Openbsd: Unix for the Practical Paranoid
    - Michael W. Lucas - No Starch Press – 1886411999
  - Freebsd Unleashed
    - Michael Urban, Brian Tiemann - Sams Publishing – 0672322064

- **Les Basics :**
  - Conception du système Unix
    - Maurice J. Bach - Dunod – 2225815968
  - Conception et implémentation du système 4.4 BSD
    - Stephen L. Nelson – Addison Wesley - 284180142X

- **Références Internet :**
  - www.ugu.com : Unix Guru Universe (infos pour l'administration)
  - www.sun.com/bigadmin/docs : Docs online sur l'administration Solaris
  - www.sysadminmag.com : Articles sur l'administration Unix
  - www.tldp.org : Linux Documentation Project

J'aime lire