

Administration de Systèmes UNIX

Jérémy Briffaut

Version original :
Mathieu BLANC

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
  - Répondre aux questions diverses des utilisateurs
  - Tuner et surveiller les systèmes
  - Assurer la sécurité
  - Mettre à jour le système (patch, upgrade, update)
  - 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
  - Assister à des réunions
  - Écrire des scripts pour automatiser un maximum de choses

Définitions et Conseils

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

- **Quelques Conseils**
  - Surtout réfléchir

  - Faire les choses pas à pas
  - Changer les choses par étapes et s'assurer que tout est réversible
  - Tester, tester et tester

  - Toujours essayer de savoir comment ça marche
  - Toutes les documentations sont vos amies
    - Man, Info, Howto
    - Magazines, Livres
    - Internet : sites, forums, news
  - Les scripts sont, aussi, vos amis

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

- **ATTENTION A L'ADMINISTRATION car ...**
- **ADMINISTRATEUR = ROOT**

- **ROOT peut TOUT FAIRE et a TOUS LES DROITS**
- **Et il est très convoité par les PIRATES**
- **Il faut, donc, sécuriser l'accès à ROOT**

  - Restreindre l'accès à ROOT (TTY, Réseau)
  - Toujours verrouiller la station (graphique ET console)
  - Tracer la connexion à ROOT et son utilisation (SU et SUDO)

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

- **L'administration, c'est aussi le contact avec les UTILISATEURS**
- **ATTENTION à ne pas devenir un BOFH**
- **Le dialogue est important ...**
- **Il faut être à l'écoute des utilisateurs ...**
- **Mais il faut savoir être très ferme avec ces mêmes utilisateurs**

- **Pour communiquer avec les utilisateurs, utiliser tous les moyens**
  - Mails
  - News
  - *write*
  - *wall*
  - *motd*
  - *issue*

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

- **Pour en finir avec les généralités ...**
- **Se méfier des outils d'administration**

- **Ils sont pratiques, simples et rapides**
- **Mais ils sont tous différents d'un système à l'autre**
- **Et quand le système sera totalement planté, ils ne fonctionneront plus**

- **Il est, donc, très intéressant de les utiliser, mais ...**
- **Il faut toujours savoir**
  - Comment ils fonctionnement
  - Quels fichiers de configuration sont modifiés
  - Comment modifier ces fichiers à la main pour le jour où tout ira mal

Définitions et Conseils

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 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 systèmes de fichiers, les processus et la mémoire virtuelle
    (ou comment tuner un système)

PLAN (enfin ...)

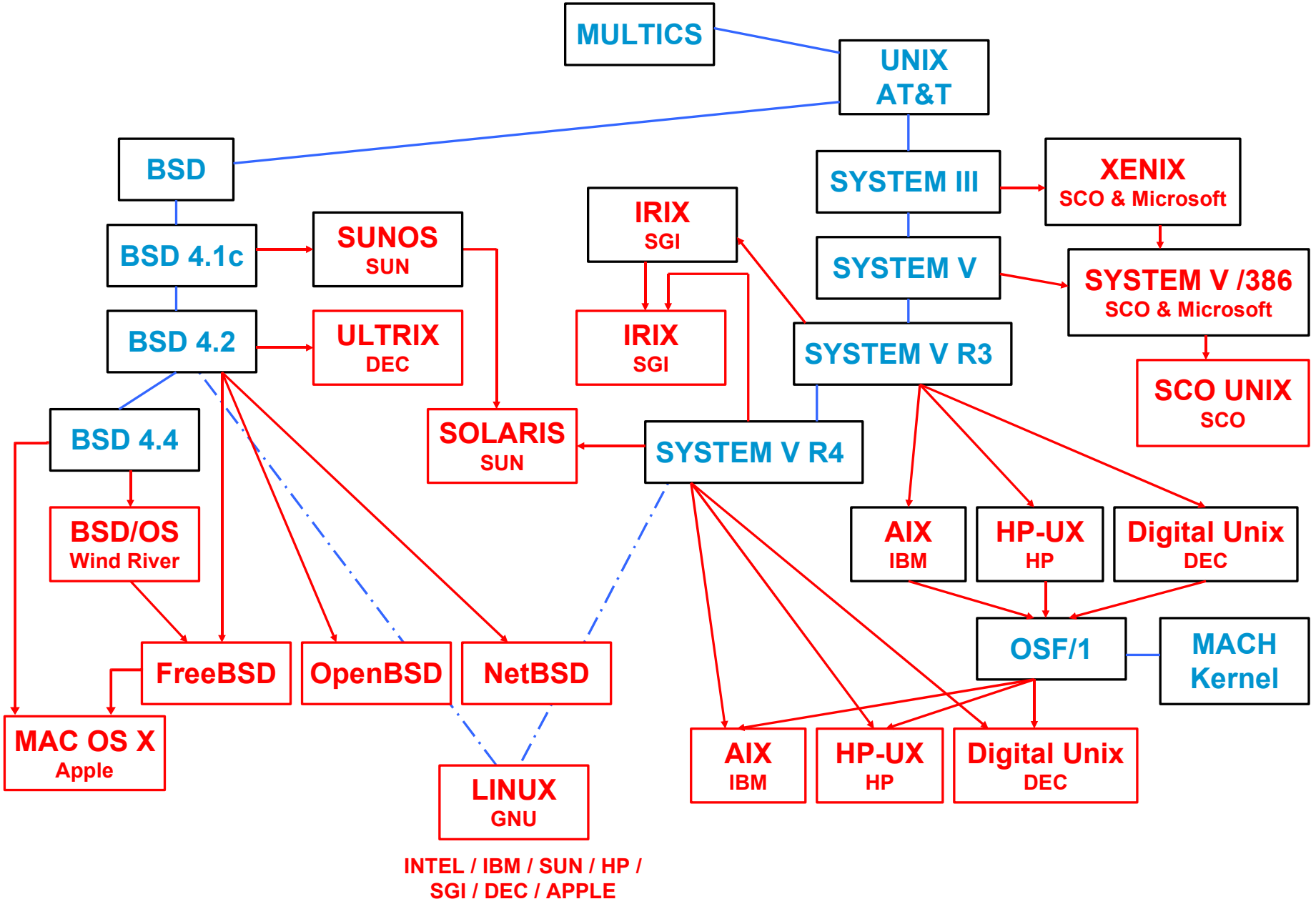
- **Exploitation d'un système Unix**
  - Les comptes utilisateurs (ou comment autoriser l'utilisation du système)
  - L'authentification (password, groups, shadow et PAM)
  - Les permissions sur les fichiers, les quotas disque et les ACLs (ou comment contrôler qui accède à quoi)
  - Exécution décalée : 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 et Accounting (ou comment jouer à Big Brother)

Programme

Historique des UNIX

Comment y voir un peu plus clair ?

Historique des UNIX



Il était une fois ...

Concepts de Base de l'Administration Système

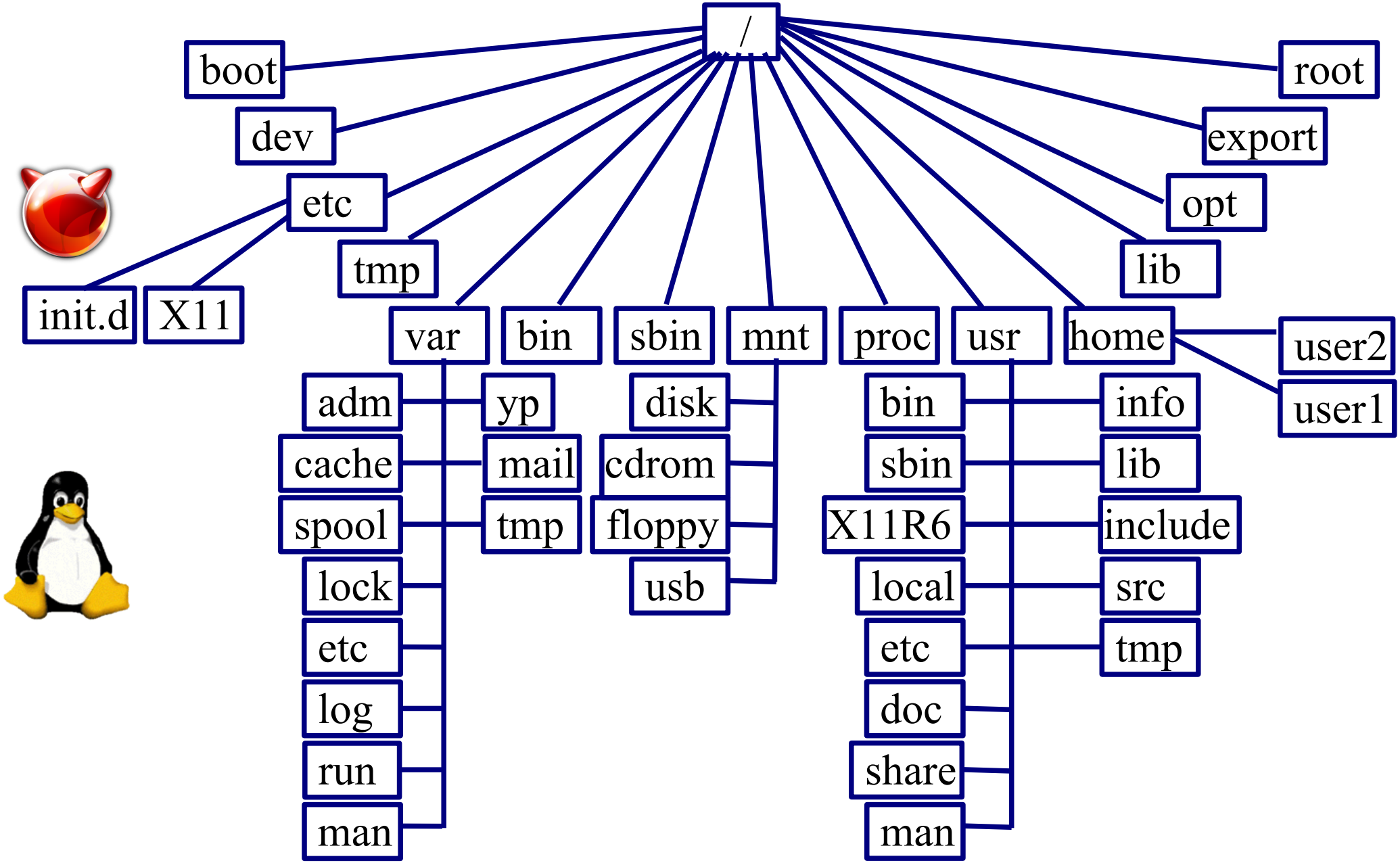
Arborescence UNIX

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, ...)

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

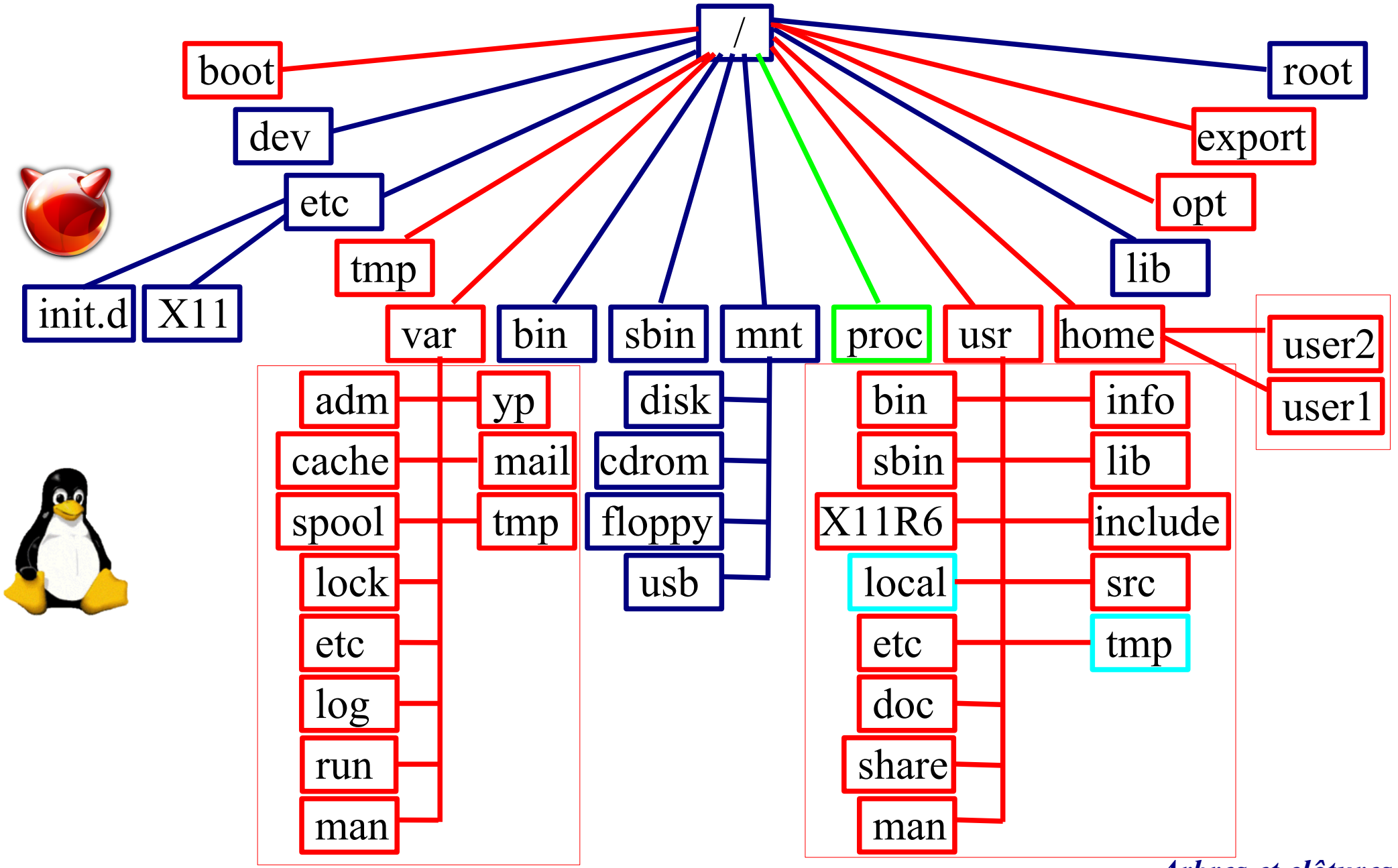
- lost+found : i-node non recouvrable lors d'un fsck

A quoi ça sert ?

Concepts de Base de l'Administration Système

Partitions, devices, RAID et LVM

Arborescence et Partitions



Arbres et clôtures

Arborescence et Partitions

- **Les répertoires souvent placés sur des partitions séparées**
  - /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

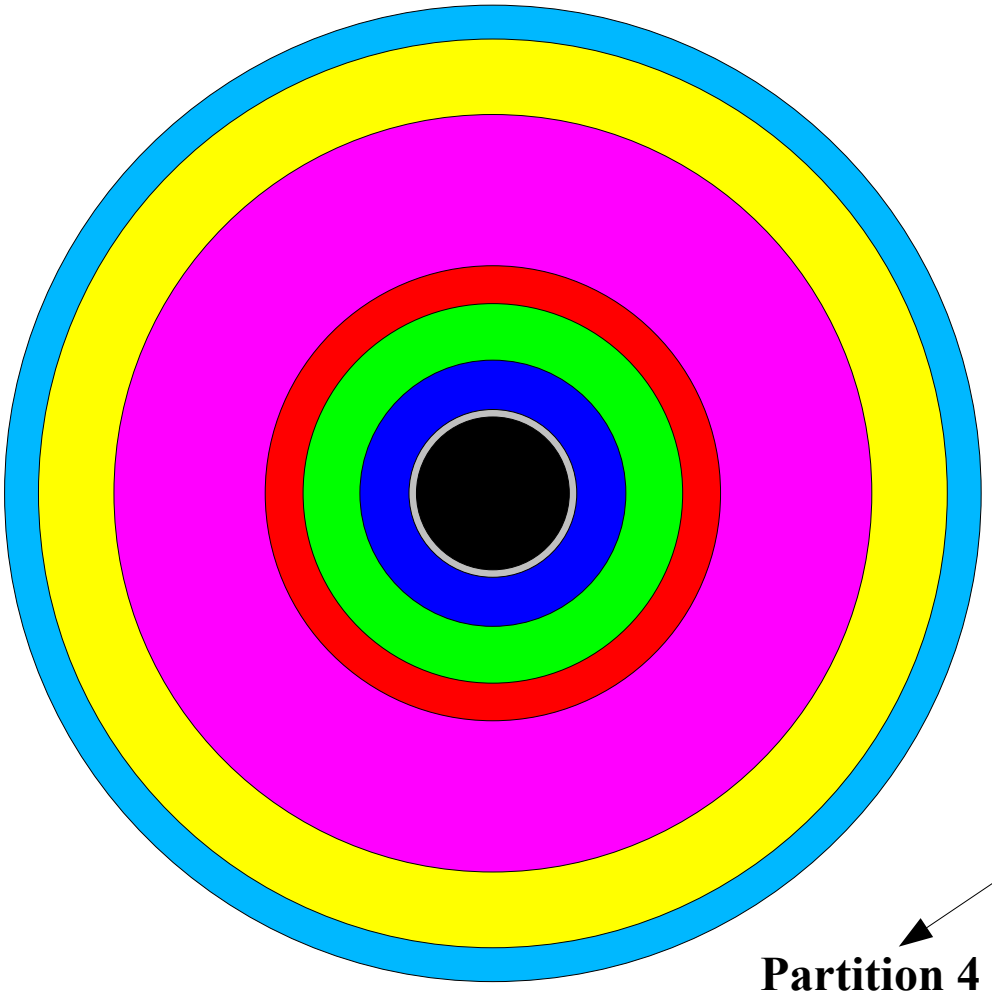
Découpons les disques

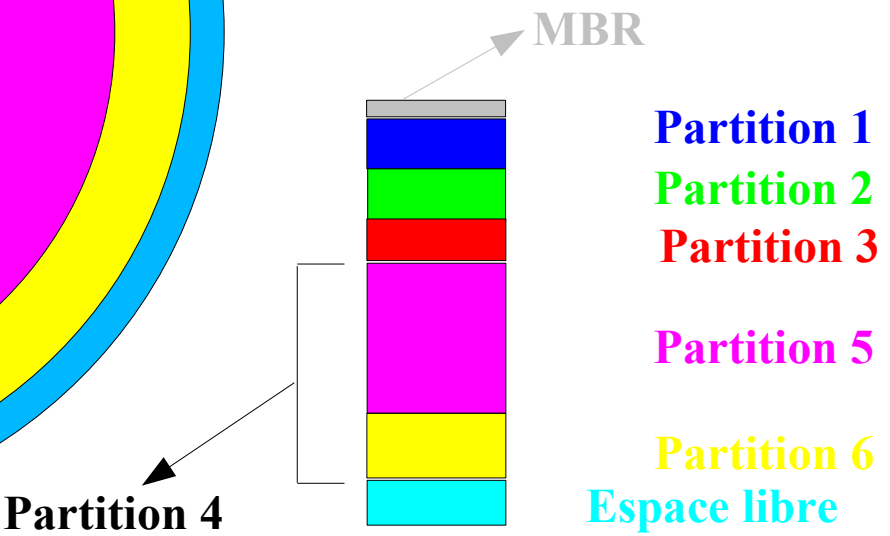
Le partitionnement DOS

Partitions primaires
- Partitions 1,2 et 3

Partition étendue
- Partition 4

Partitions logiques
- Partitions 5 et 6



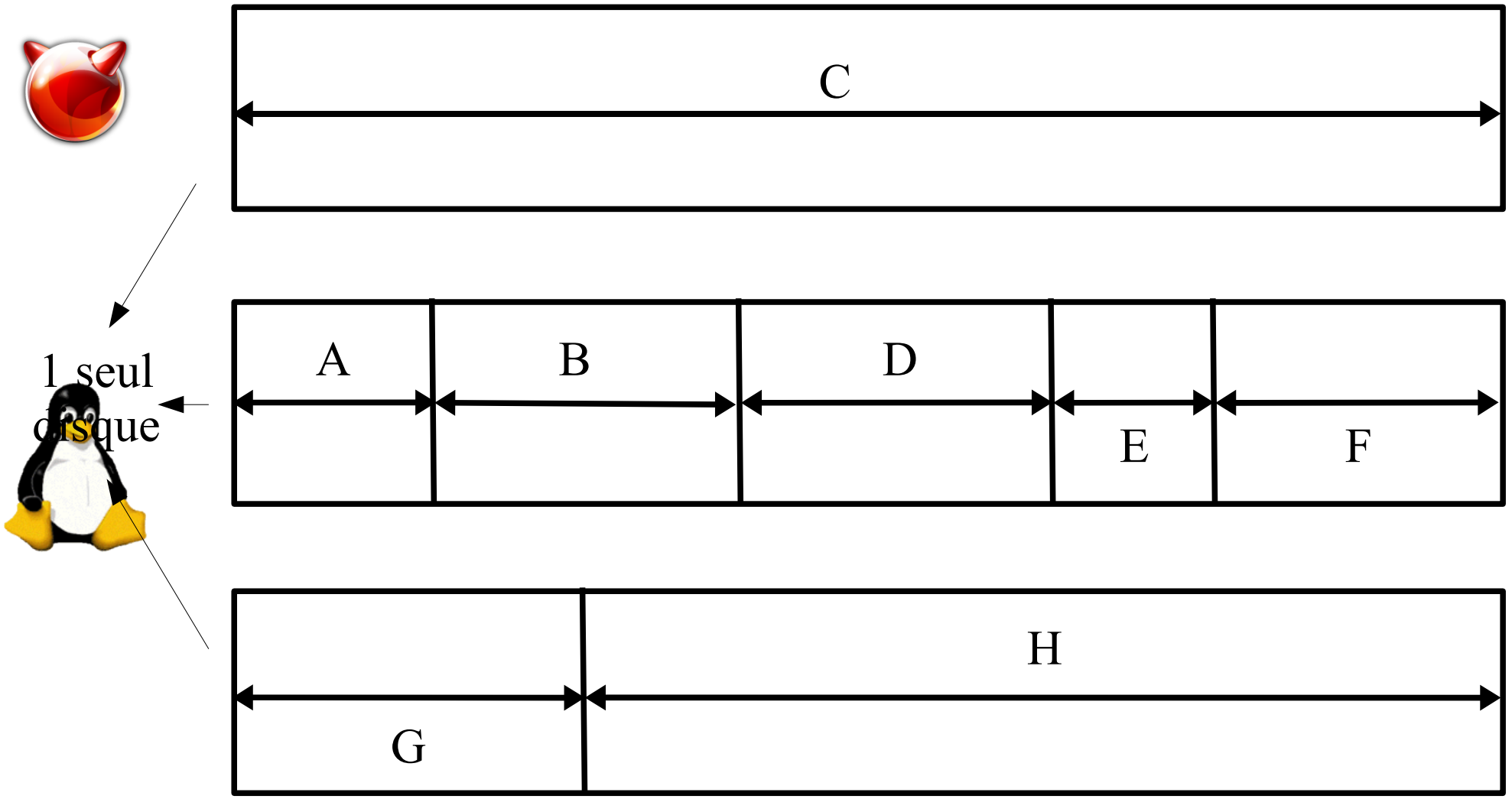


PC et Dos

Partitions

- **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 est censé contenir (attention, il ne correspond pas forcément au système de fichiers réellement en place)
  - 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**

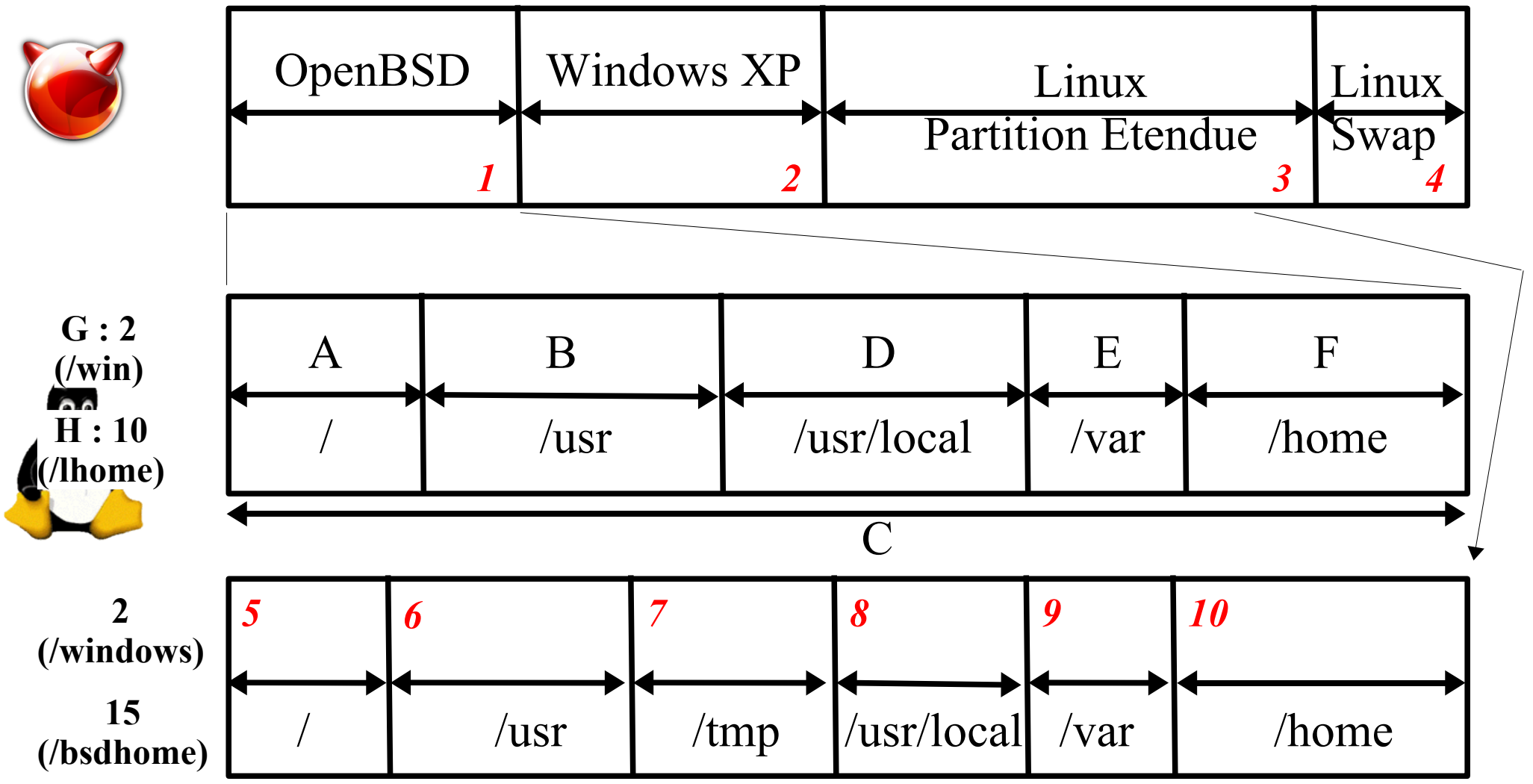
Partitions

- **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 installer 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 partitionnée 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 : sd [a|b|c|...] [1|2|3|4|...]
    - a : premier device vu sur les bus scsi
    - b : deuxième device vu les bus scsi
    - 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)

Partitions

- **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, 4 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

Assouplir 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 *<u>/dev/dsk/c2t0d0s3</u>* (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

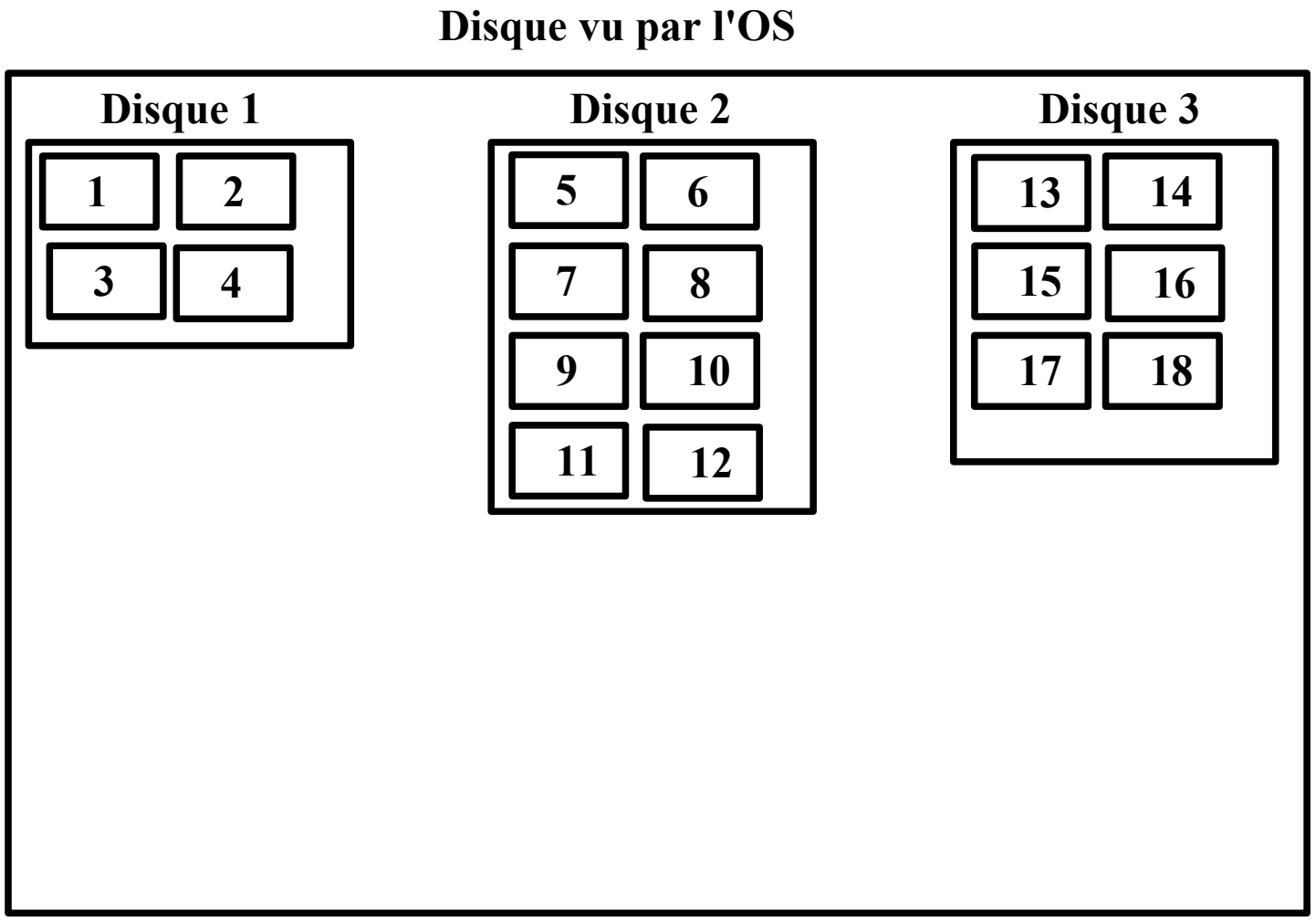
- **RAID logiciel se compose**
  - De la couche logiciel dans le noyau de l'OS
  - De disque durs connectés au système (SCSI ou IDE)
  - On peut trouver des baies SCSI, vu par l'OS, comme autant de disques durs

- **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**

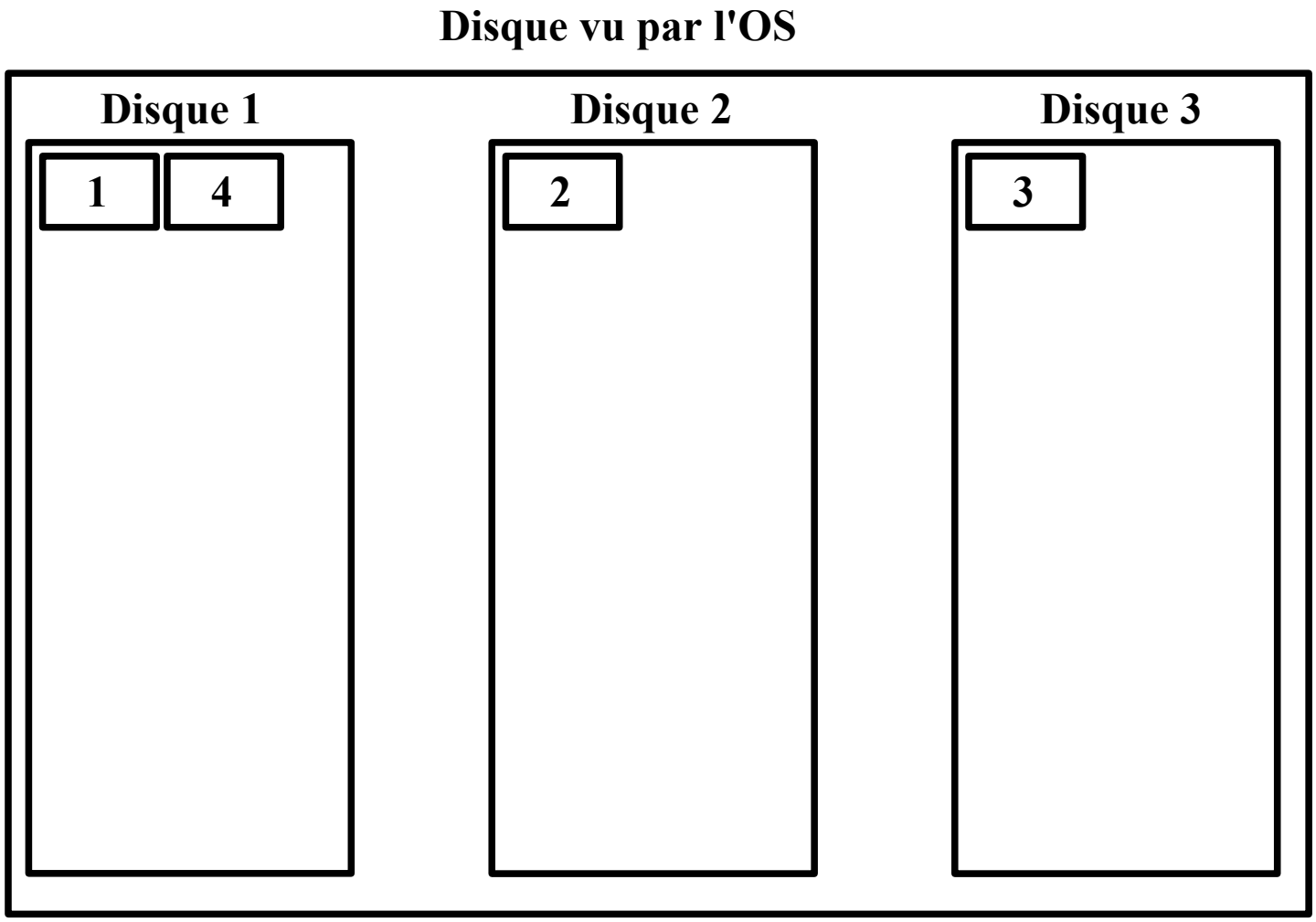
Assouplir le RAID

- **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



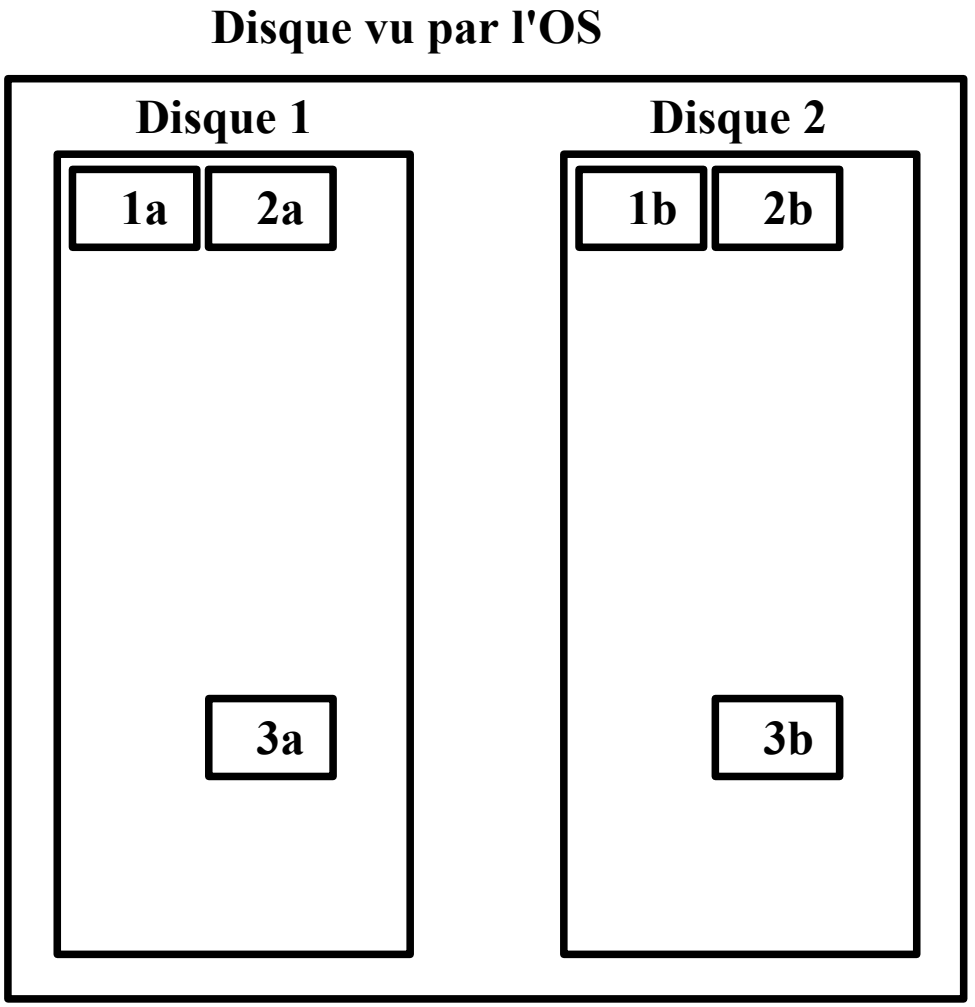
Linéaire

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



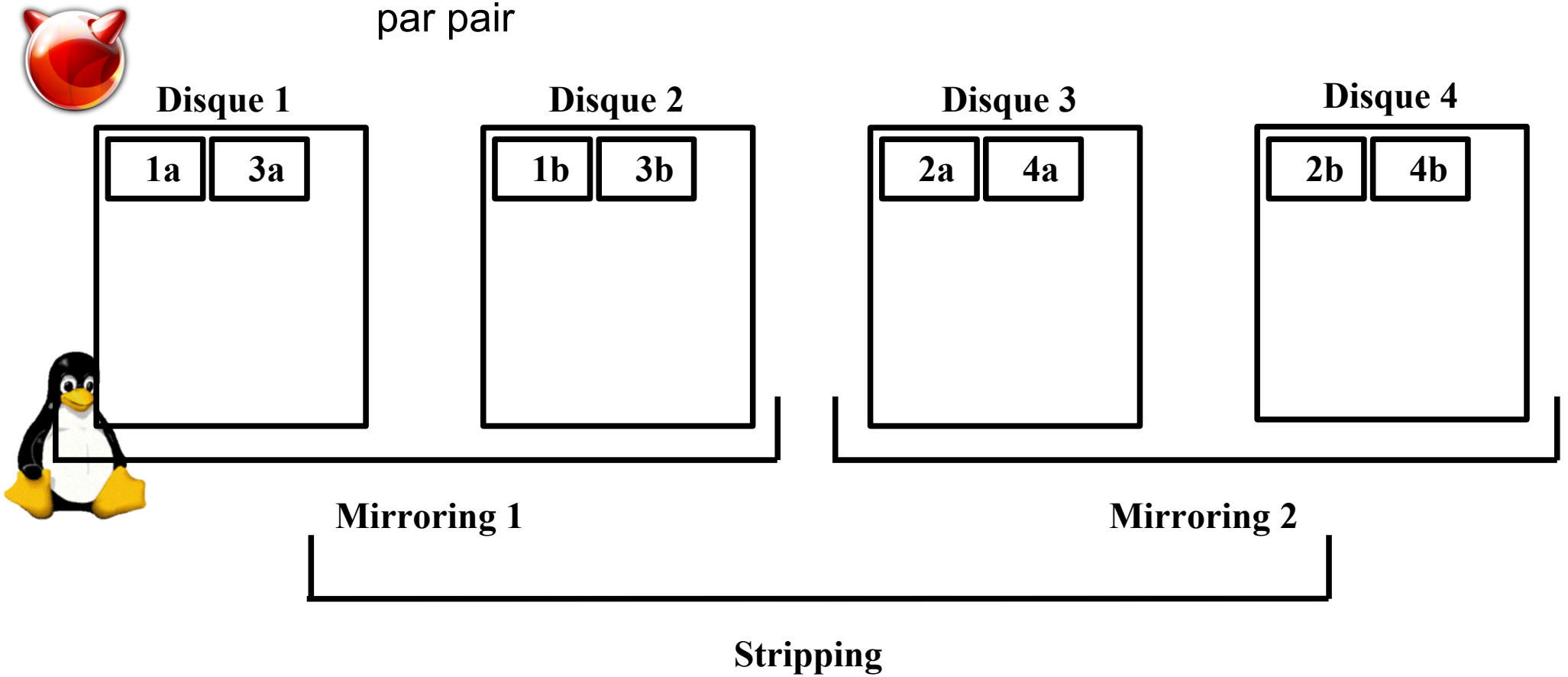
RAID-0

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



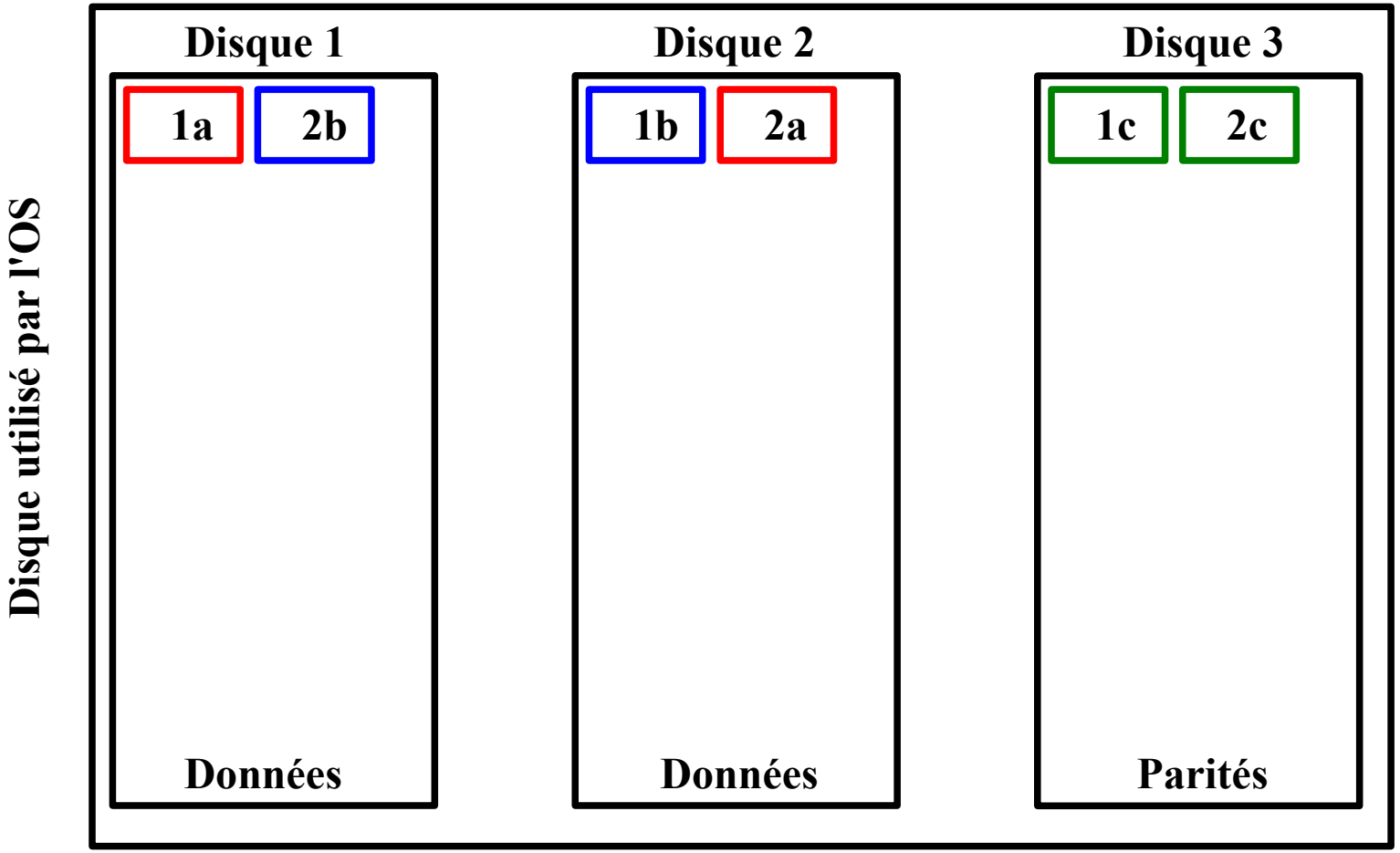
RAID-1

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



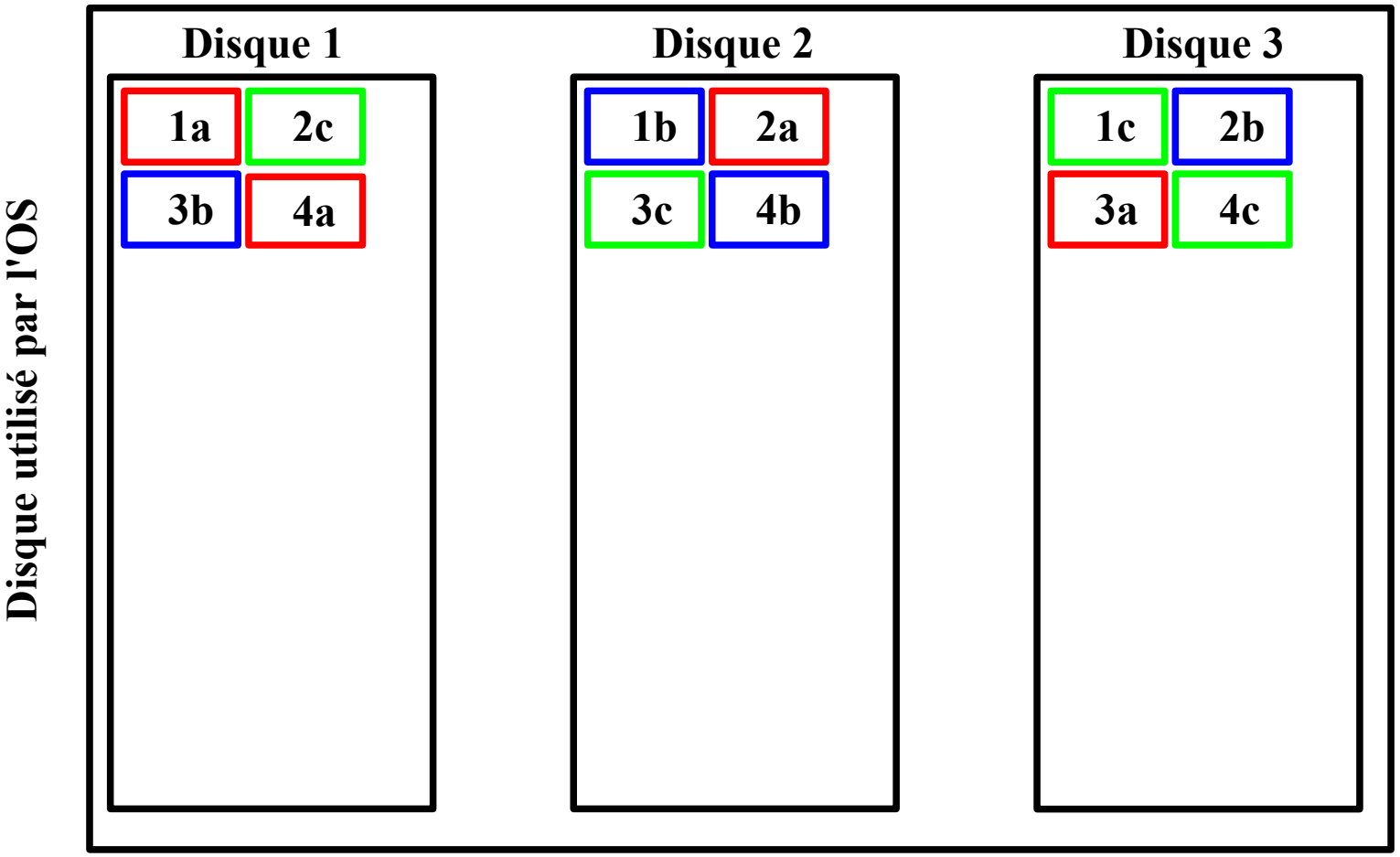
RAID-0+1=?

- **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-7
    - N'est pas un standard : propriétaire (uniquement hardware)
    - Basé sur le RAID-3 et RAID-4 en améliorant les performances en lecture et écriture par un système de caches

Les autres RAIDs

- **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

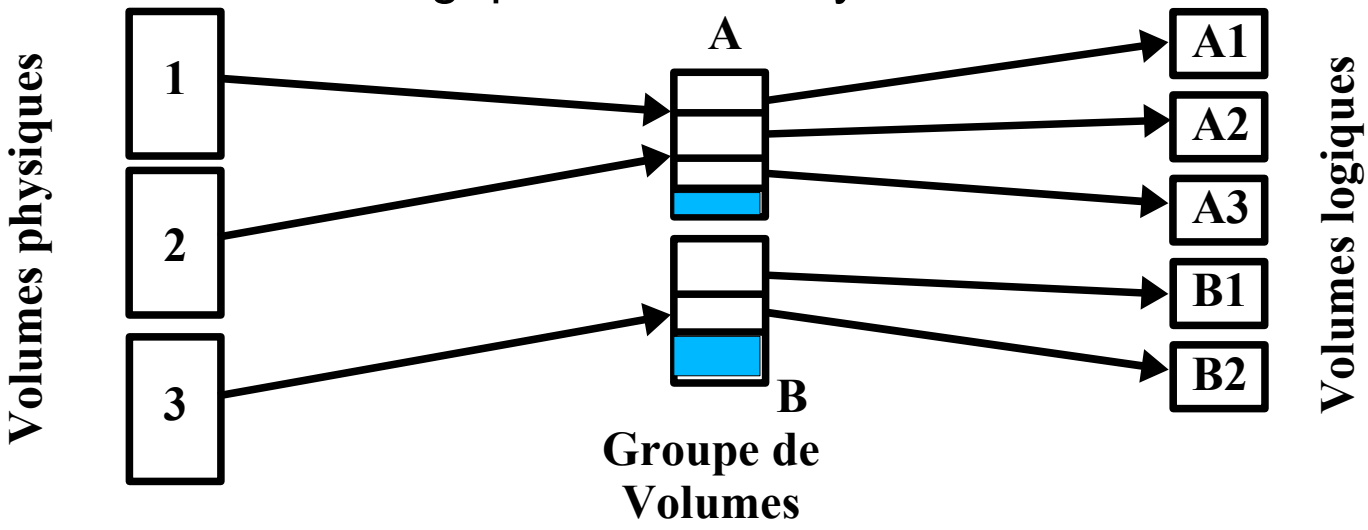
- Permet :
  - De s'affranchir des tailles physiques des disques durs
  - D'augmenter dynamiquement la taille d'un système de fichiers
  - D'ajouter ou enlever un disque dur sans devoir tout re-installer
  - De créer des snapshots pour assurer des backups cohérents

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)
  - Un groupe de volumes va permettre de définir l'équivalent du disque physique, c'est dans un groupe de volumes que l'on va définir les partitions pour créer les systèmes de fichiers
  - 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 couches du LVM

- **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 ?
  - Est-il possible d'avoir du LVM pour / ?

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

Partitions et Arborescence

- **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)

fstab

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 périphérique
    - 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, ...)**

Création de systèmes de fichiers

- **Mais certains systèmes de fichiers n'ont pas besoin d'être créés**
- **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)

Quizz

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

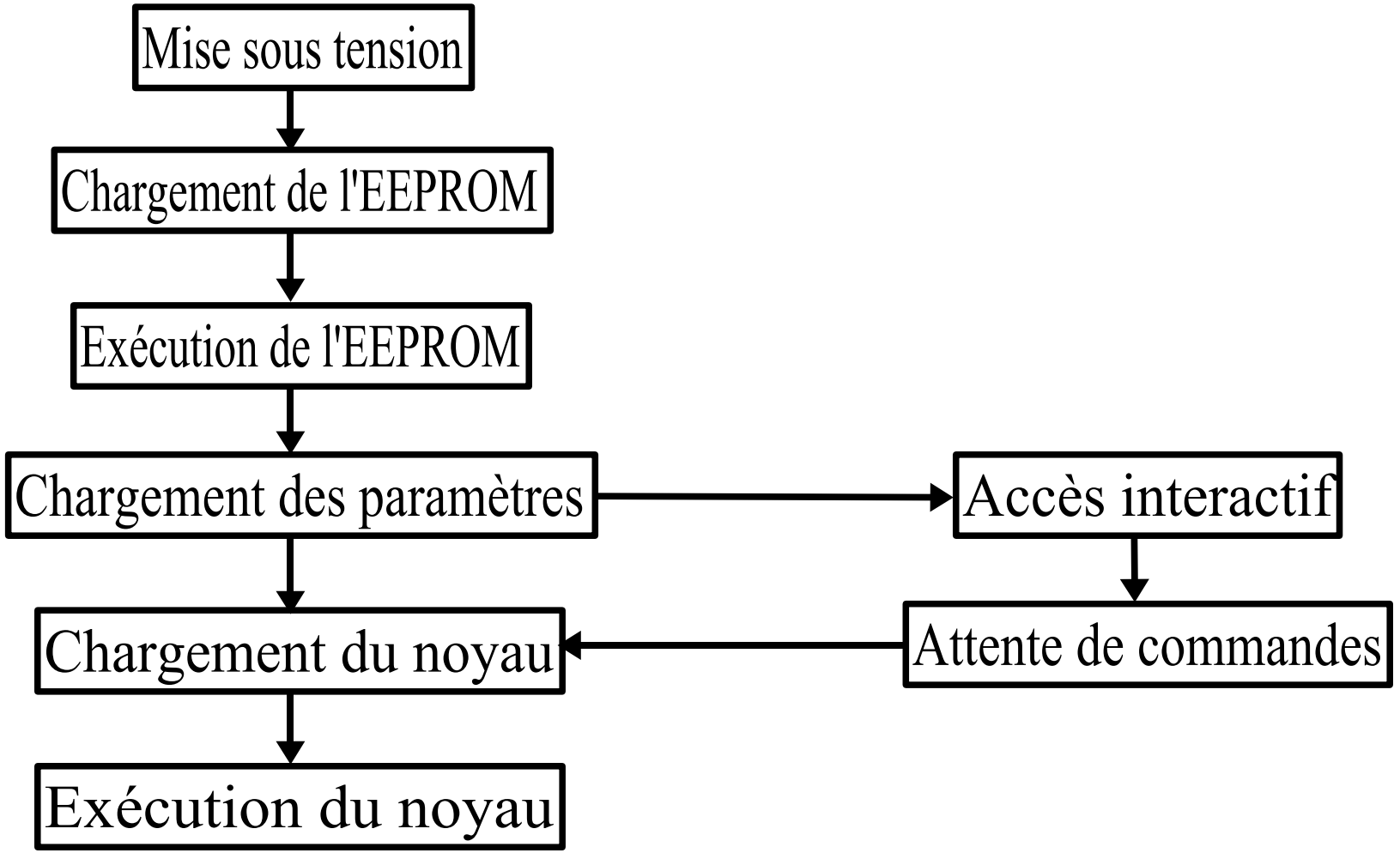
- **Qu'est-ce qu'un boot loader ?**
  - Pour qu'un système Unix puisse se lancer, il faut que le noyau soit chargé en mémoire et qu'il s'exécute
  - Le but principal d'un boot loader est de chargé le noyau en mémoire et de le lancer
  - Un boot loader peut permettre, aussi, de
    - Choisir entre plusieurs 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 plate-forme pour exemple :
    - Sun (Solaris)
    - PC (Linux)

Comment ça peut booter ?

Boot Loader

- **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 :**



Boot Loader

- **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

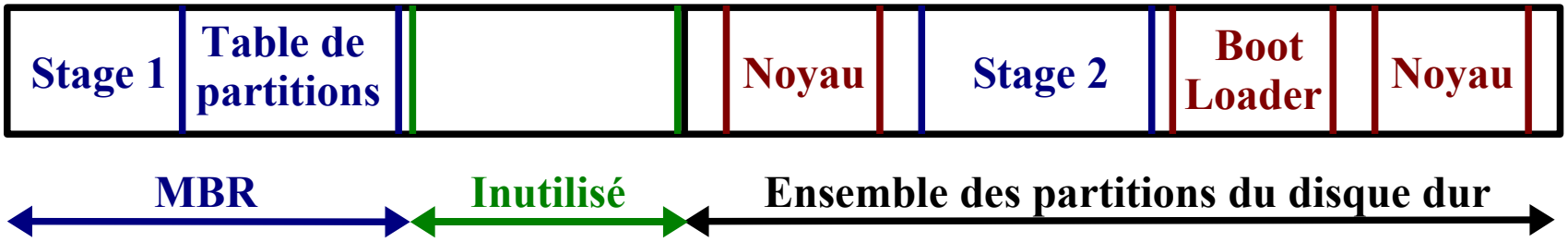
Boot Loader

- **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

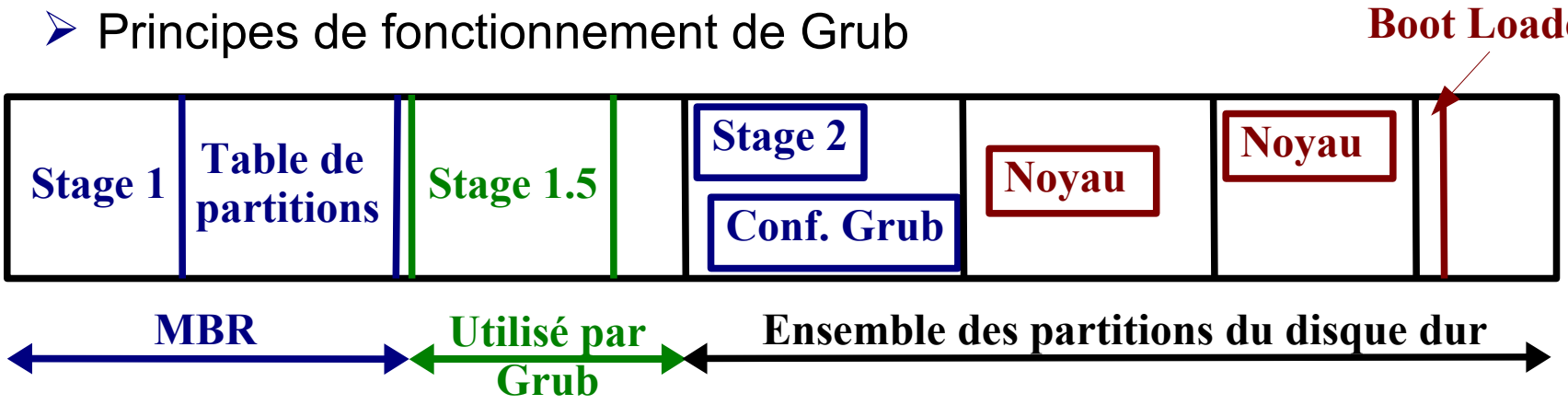
    - GRUB
      - Boot loader GNU
      - Très flexible
      - Permet de booter autre chose que Linux
      - Possède une gestion interactive
      - Permet la lecture des systèmes de fichiers
    - Pour ces 2 boot loaders, on retrouve les 2 stages qui sont chargés l'un après l'autre

- **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



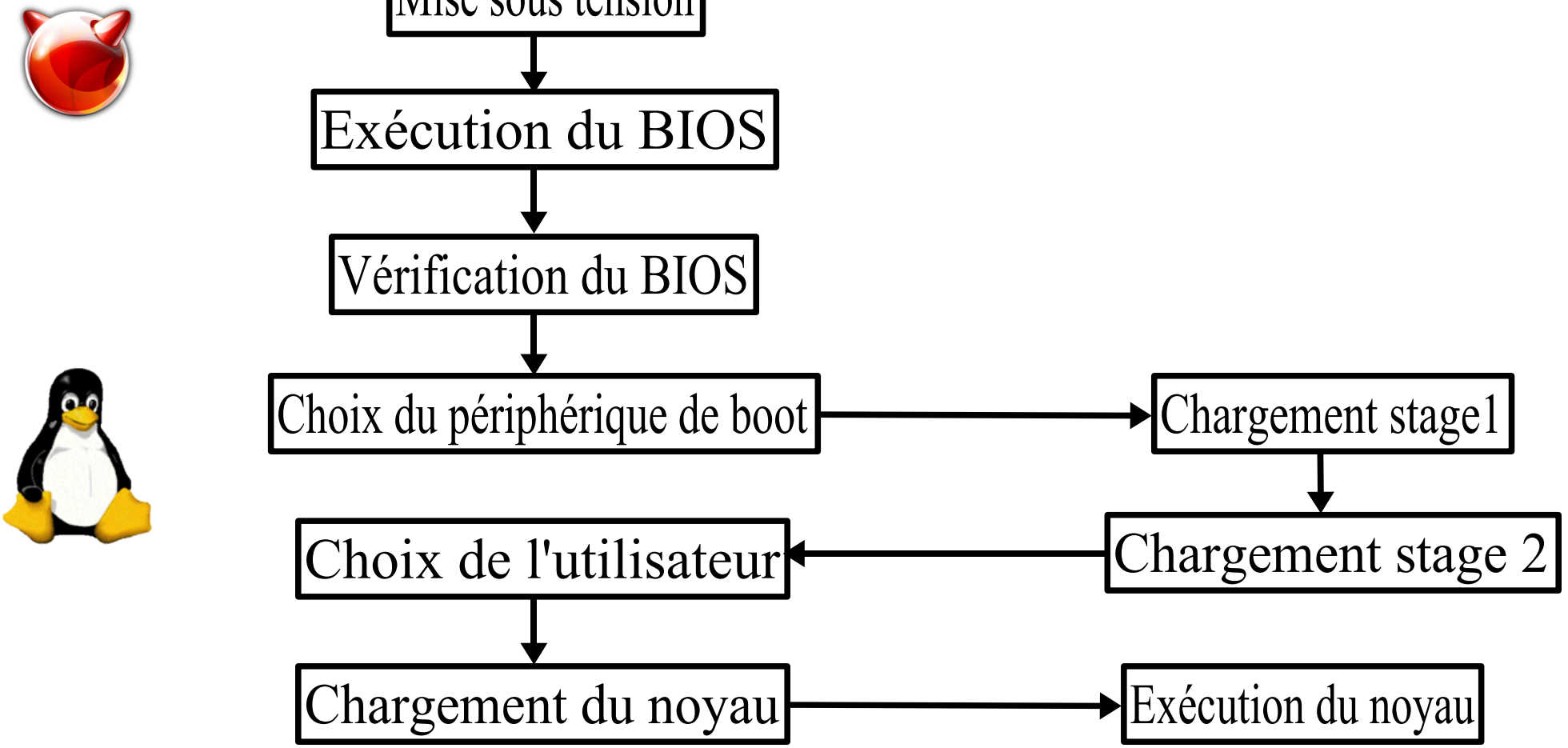
    - A l'installation de Grub, les stages 1, 1.5 et 2 sont copiés
      - Le stage 1 est installé pour charger le stage 1.5
      - Le stage 1.5 permet de lire un système de fichiers spécifique et charge le stage 2 qui est vu comme un fichier
      - Le stage 2 permet de lire le fichier de configuration 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

Boot Loader

- **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**

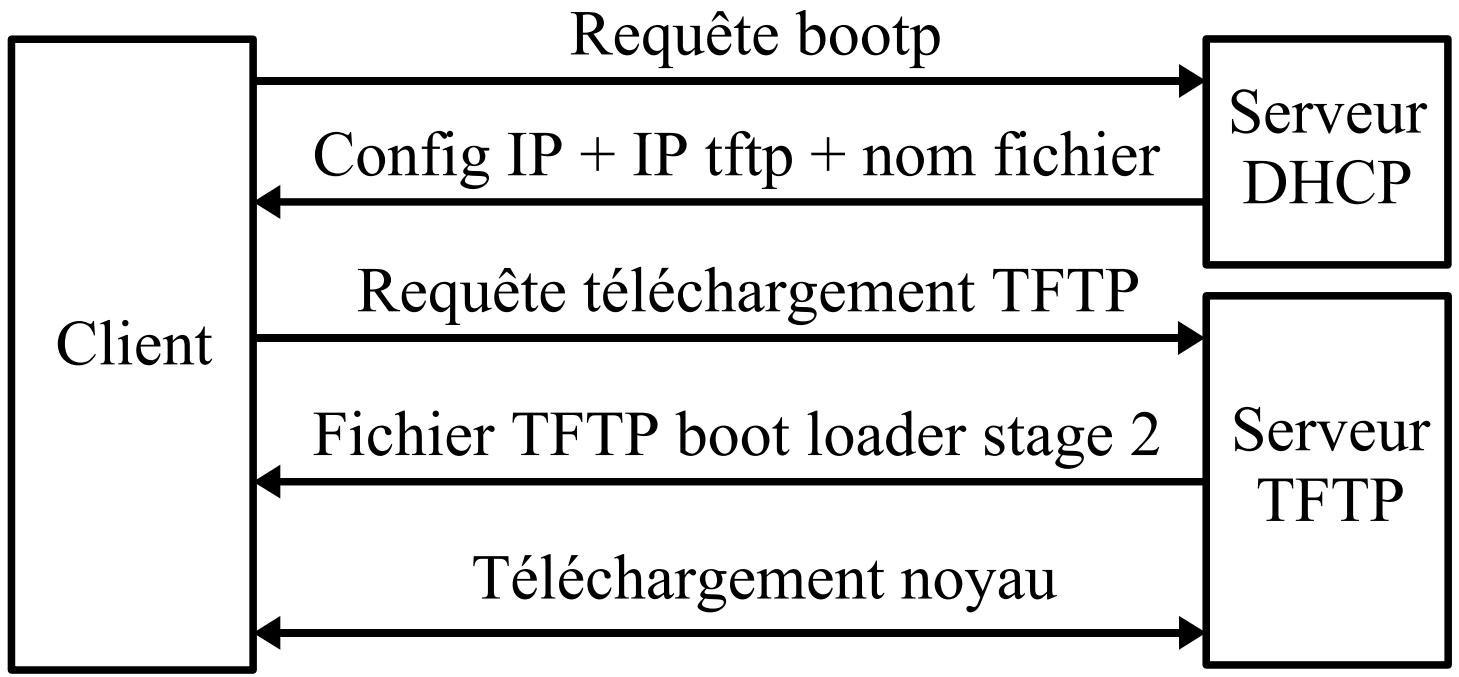


- **Sur SUN et sur PC, il existe d'autres méthodes de boot**
  - Par le réseau
  - Sur CD-ROM
  - Sur bandes de types DAT ou DDS

- **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



Boot Loader

- **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 possible 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 initialise tous les périphériques de la machine
- Si un initrd a été chargé par le boot loader, il exécute le script linuxrc
- 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
- Le lancement d'init termine 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 ...

Démarrage des Systèmes Unix

- **Init terminera la procédure de démarrage selon qu'il est BSD ou SYSV**

- **Pour les BSD**

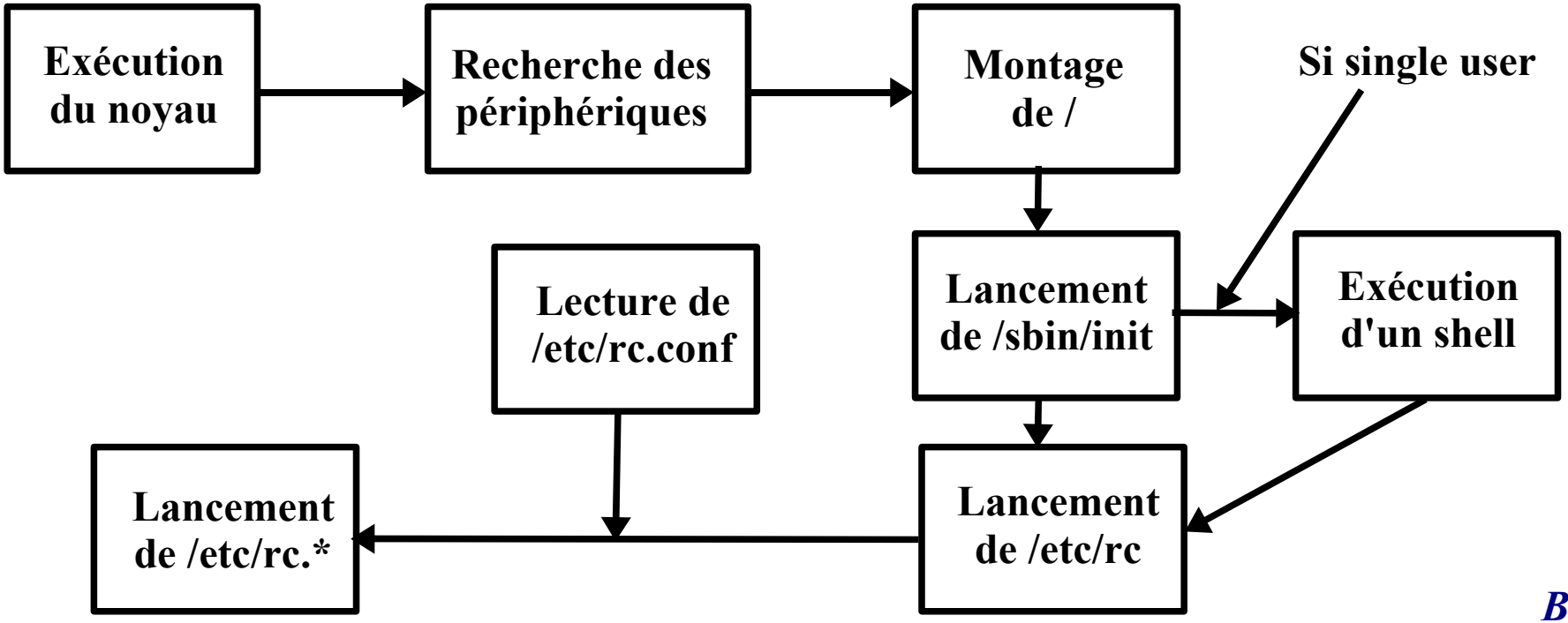
  - init cherche le script /etc/rc et l'exécute
  - Le script /etc/rc va exécuter d'autres scripts dans /etc et commençant par rc (rc.boot, rc.local, rc.net, ...)
  - Le script /etc/rc et les scripts lancés utilise le fichier /etc/rc.conf qui regroupe les variables définissant le comportement des scripts
  - 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

- **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

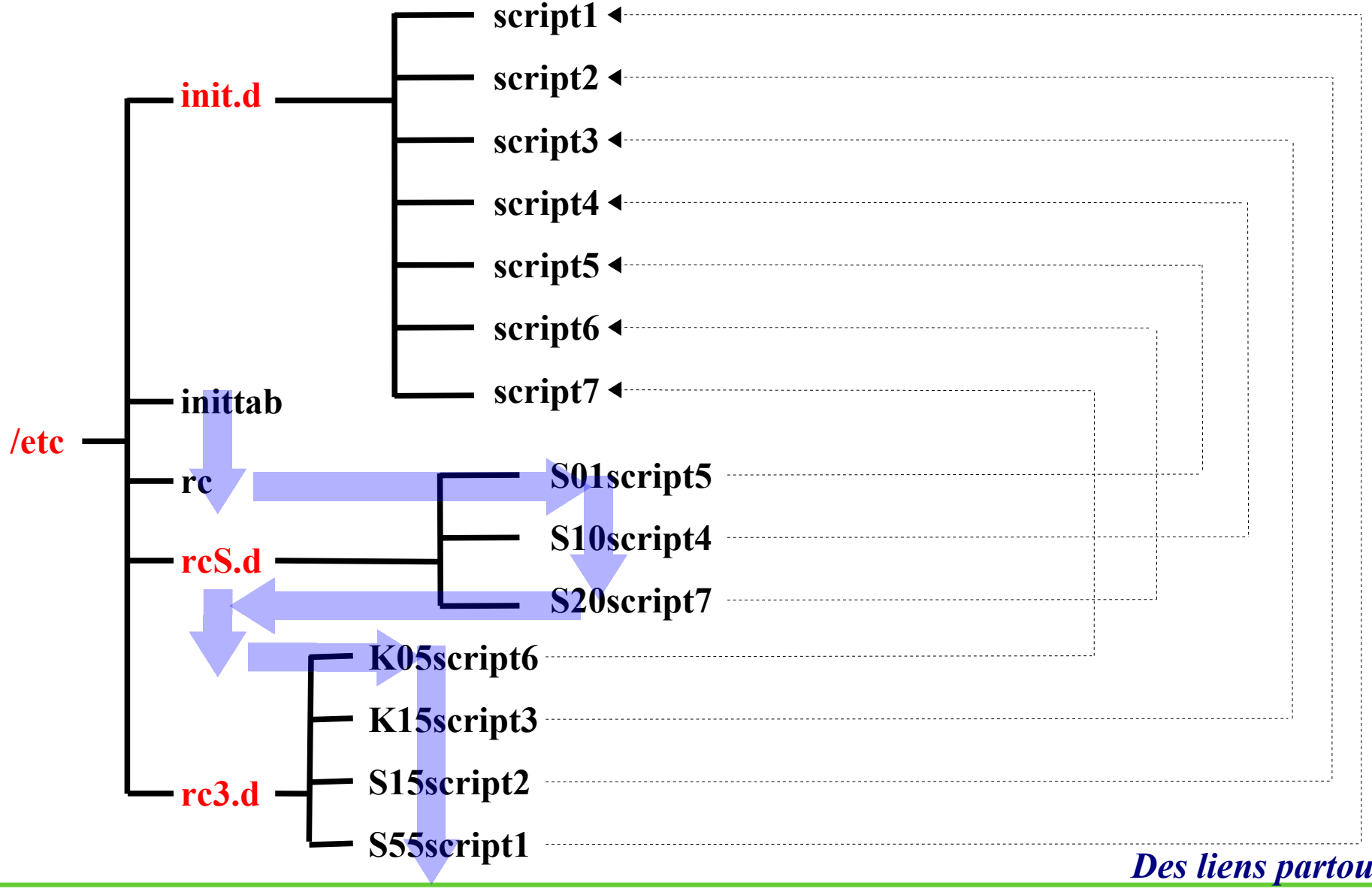
- **Pour les SYSV**
  - Un seul script est présent dans /etc/init.d, à la fois pour le démarrage et l'arrêt d'un service, mais il doit prendre en argument **start** ou **stop**

  - 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

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

Arrêt des Systèmes Unix

- **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 ?**

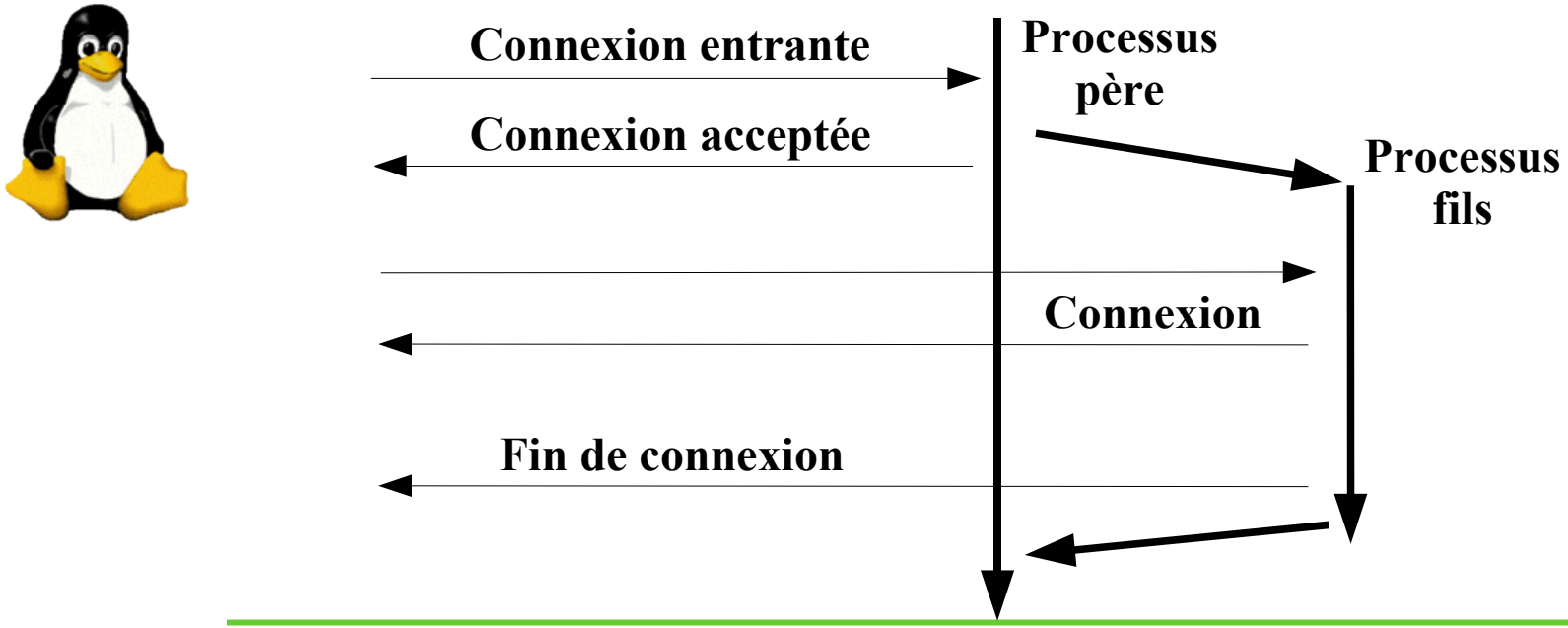
- **Le mot « démon » vient de « daemon » qui, en anglais, désigne une divinité immortelle (et non un démon au sens « devil »)**
- **Un démon est un processus qui ne meurt jamais, qui ne s'arrête jamais**

- **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 parmis 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 ?

Démons et Lancement de Services

- **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

Démons et Lancement de Services

- **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