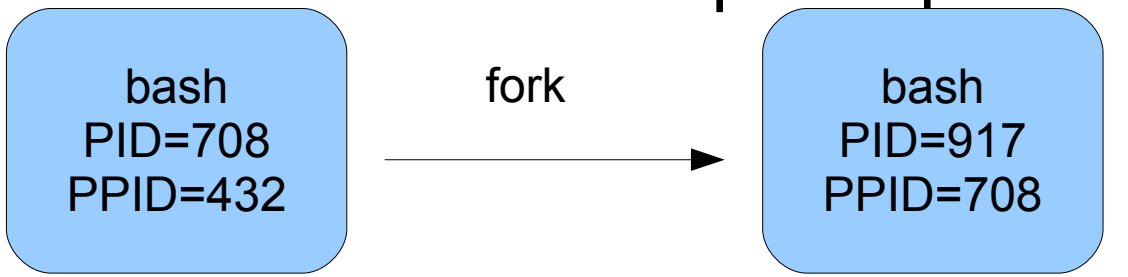
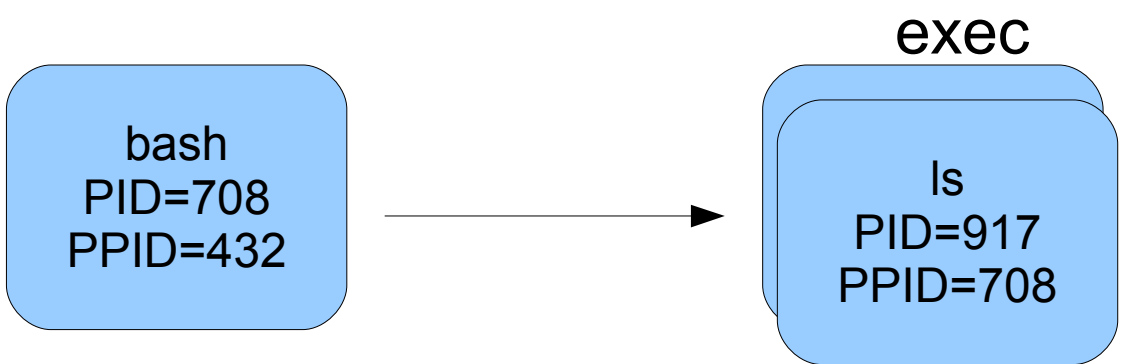
processus

- un programme: un fichier sur disque
- un processus: un programme en cours d'exécution
  - le code exécutable du programme
  - les données de l'instance en train de s'exécuter
- programme réentrant:
  - deux instances du même programme partagent le même code exécutable
  - elles ont par contre chacune leurs données
- processus système (daemon)/utilisateur

hiérarchie de processus, recouvrement

- un processus (processus fils) est toujours créé par un autre processus (processus père):
  - fork: création d'une copie du processus père
  - exec: recouvrement par le processus fils

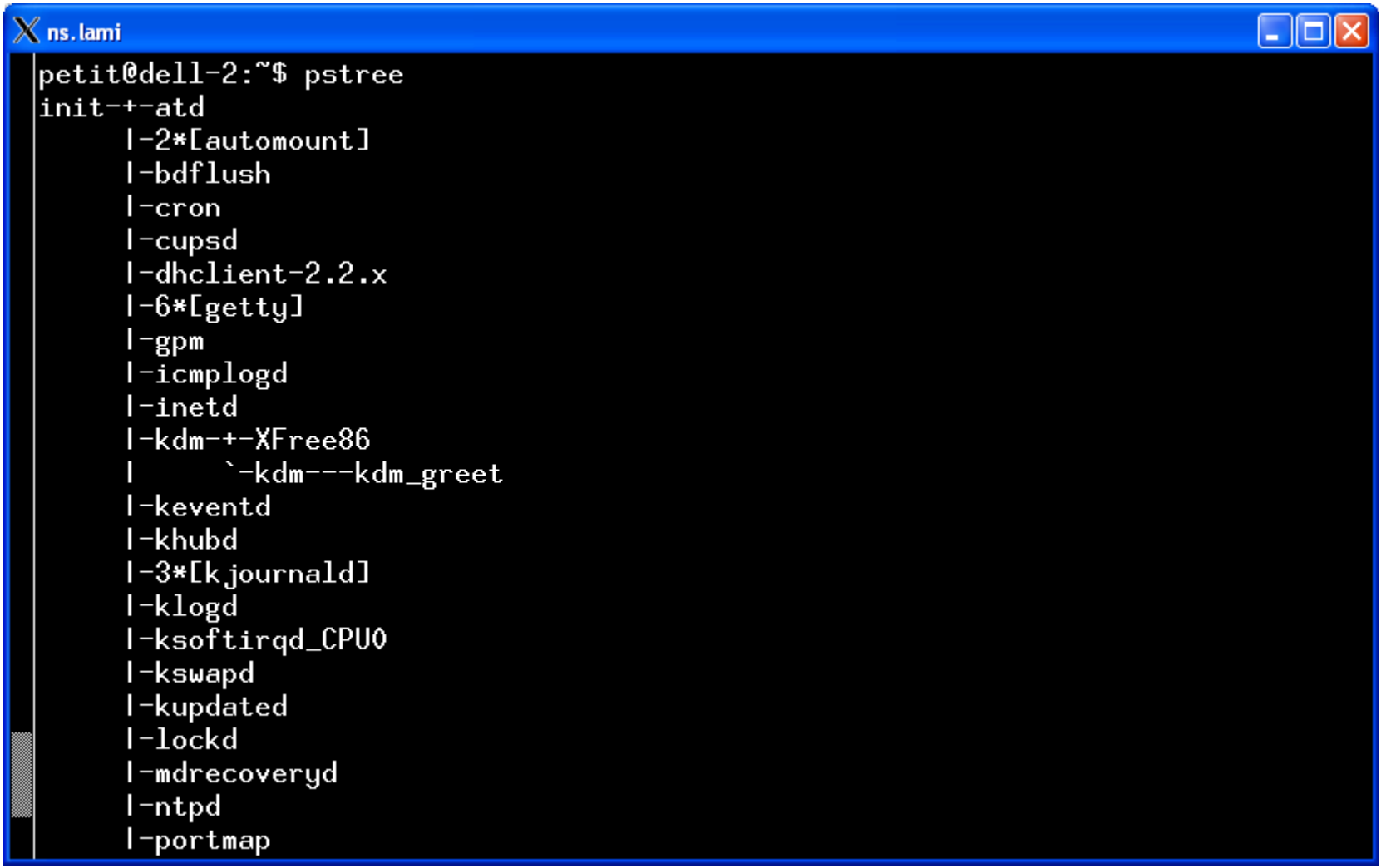




Hiérarchie de processus

- tout processus a un processus parent sauf le processus initial
- processus initial : init (pid 1)
- arrêter la machine: demander à init d'arrêter tous ses processus fils

pstree


ns.lami
petit@dell-2:~$ pstree
init-+-atd
|-2*[automount]
|-bdflush
|-cron
|-cupsd
|-dhclient-2.2.x
|-6*[getty]
|-gpm
|-icmplogd
|-inetd
|-kdm-+-XFree86
|`-kdm---kdm_greet
|-keventd
|-khubd
|-3*[kjournald]
|-klogd
|-ksoftirqd_CPU0
|-kswapd
|-kupdated
|-lockd
|-mdrecoveryd
|-ntpd
|-portmap

caractéristiques des processus

- statiques
  - PID
  - PPID
  - propriétaire réel (UID, GID)
  - terminal d'attache pour les entreés-sorties
- dynamique
  - propriétaire effectif (EUID, EGID)
  - priorité
  - nice
  - consommation cpu/mémoire
  - dossier de travail

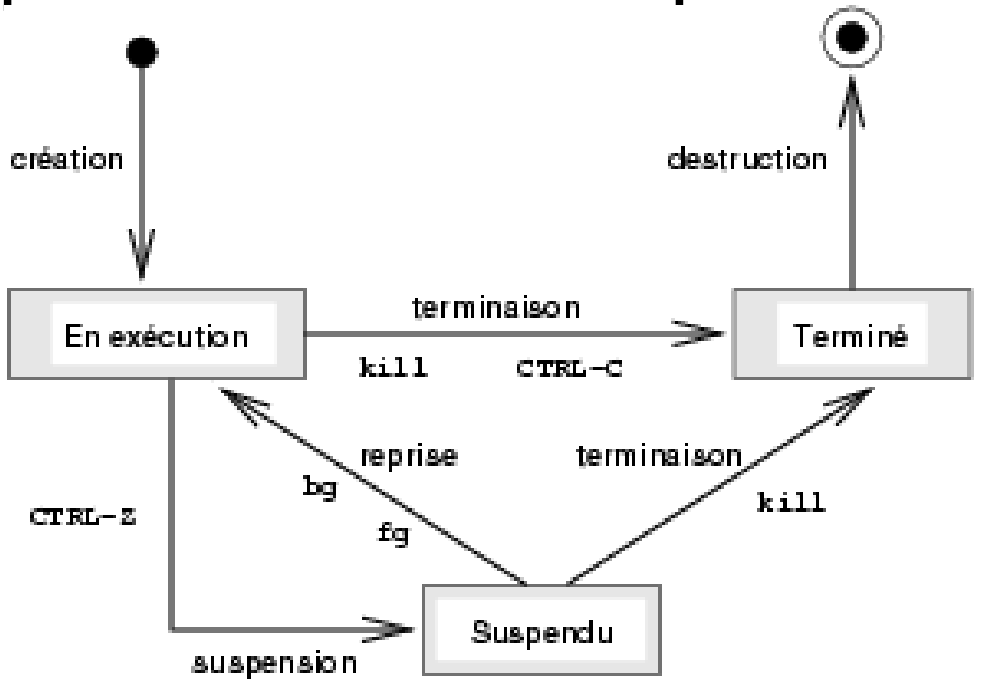
commande ps

- 2 syntaxes (Linux):
  - syntaxe Systeme V: option précédées de -
  - syntaxe BSD: options NON précédées de -
  - quelques options SysV:
    - -e ou -A: tous les processus
    - -a: tous les processus associés à un terminal
    - -H: représentation hiérarchique (forêt)
    - -f: format complet;-l: format long (encore plus détaillé)
    - -o: pour modifier le format de sortie (cf manuel)
    - -g, -p, -t, -u: n'affiche que les processus des groupe (-g), processus (-p), terminaux (-t) ou utilisateurs (-u) listés.

commande ps: exemple

Etat d'un processus

- R: exécution
- Z: zombi: il est mort mais son père ne le sait pas
- D: le processus ne peut être interrompu
- S: suspendu
- T: terminé



gestion de processus & shell

- commande & : lancement de commande en tâche de fond
- bg: reprise de commande en tâche de fond
- fg: reprise de commande en avant plan
- jobs: liste des commandes lancées
- Ctrl-C: arrêt (SIGTERM) du processus
- Ctrl-Z: suspension (SIGSTOP) du processus

Signaux

- permettent au système de communiquer avec les processus
- signaux utiles
  - STOP: suspendre
  - CONT: reprendre
  - HUP (1): souvent: relecture configuration
  - KILL(9): tuer sans possibilité de traitement
  - INT(2): équivalent à Ctrl-C: interruption gérable. permet au processus de gérer son interruption
- kill -signal PID

priorité des processus

- l'exécution des divers processus est gérée par un ordonnanceur (scheduler)
- une priorité est définie dynamiquement
- but: que chaque processus puisse avancer son exécution tout en respectant des priorités
- nice/renice: permet d'influer sur la priorité des processus
  - de 0 à 19 pour un utilisateur
  - de -20 à 19 pour root
  - plus le chiffre est élevé, moins le processus est prioritaire

code de retour

- valeur à laquelle le processus père peut accèder
- 0: terminaison normale
- autre valeur: situation anormale

systèmes de fichiers

- différents systèmes de fichiers:
  - exemples
  - journalisation
  - création d'un système de fichier
  - exemple sous linux
- partition
  - associée à un système de fichier et un point de montage
  - de swap
- gestionnaire de volume logique
  - développer LVM sous Linux

acl POSIX

- dans une version ultérieure de ce document
  - Cf TD
- cas particuliers:
  - linux debian
  - FreeBSD

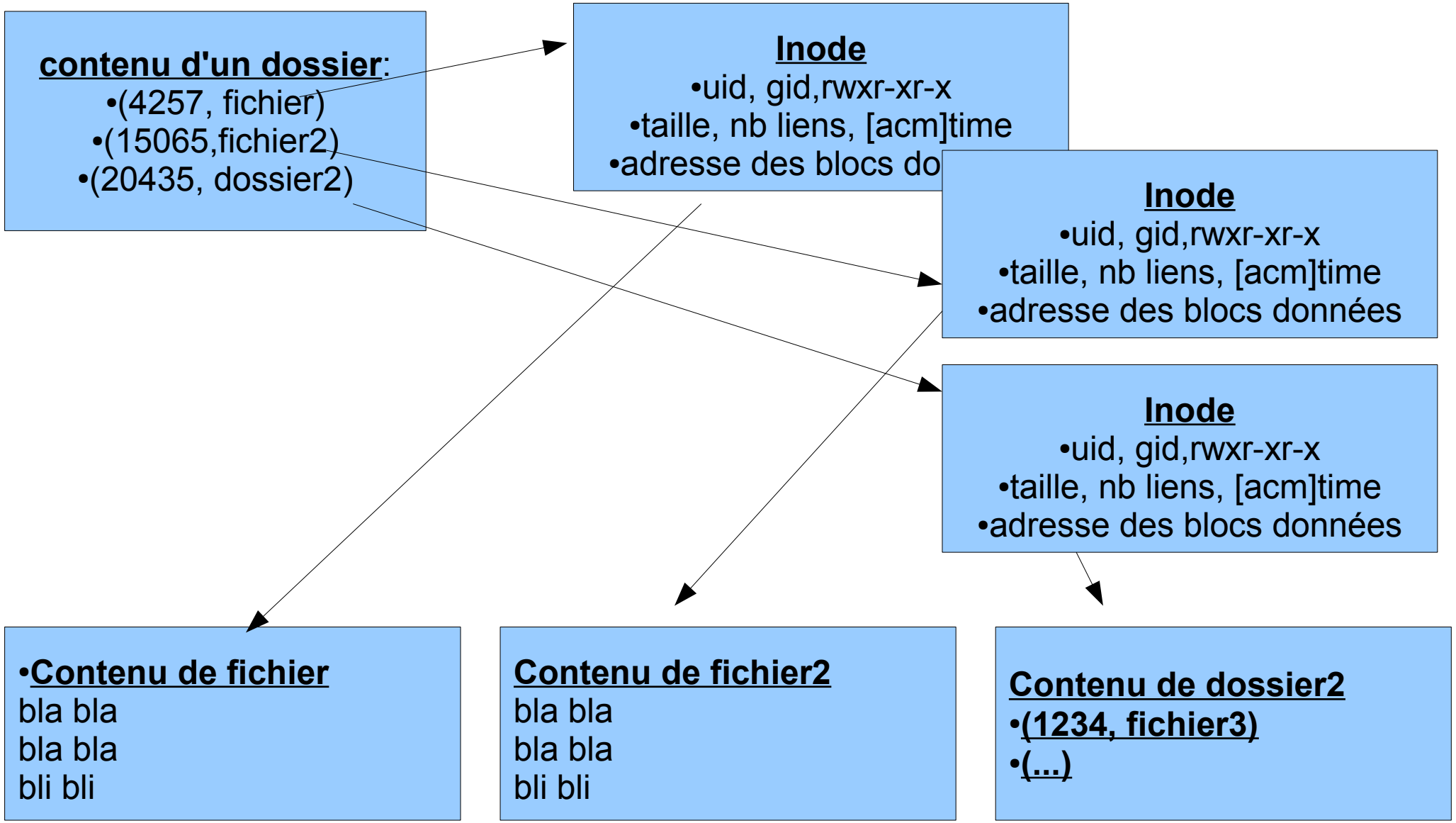
Systèmes de gestion de fichiers (SGF)

- SGF: mode d'organisation et de stockage des données sur disque;
- Exemples: FAT32, NTFS, ext2fs, ext3fs, reiserfs, UFS, ...
- Les SGF ont des propriétés et fournissent des services variés
- Exemple:
  - les SGF Unix (ext2fs, UFS, ...) : droits sur les fichiers.
  - FAT32: pas de droits d'accès aux fichiers

SGF (suite)

- Les SGF unix fournissent un sous-ensemble commun de fonctionnalités: celui dont nous parlerons.
- Chaque SGF peut fournir plus que ce sous-ensemble
- Fichier unix: fichier disque mais aussi ressource du système (pilote de périphérique, terminal, mémoire, ...)
  - /dev/hda1 : partition 1 du disque 1 (Linux)
  - /dev/kmem: mémoire virtuelle du noyau

Fichiers


contenu d'un dossier:
•(4257, fichier)
•(15065,fichier2)
•(20435, dossier2)
Inode
•uid, gid,rwxr-xr-x
•taille, nb liens, [acm]time
•adresse des blocs do
Inode
•uid, gid,rwxr-xr-x
•taille, nb liens, [acm]time
•adresse des blocs données
Inode
•uid, gid,rwxr-xr-x
•taille, nb liens, [acm]time
•adresse des blocs données
•Contenu de fichier
bla bla
bla bla
bli bli
Contenu de fichier2
bla bla
bla bla
bli bli
Contenu de dossier2
•(1234, fichier3)
•(...)

SGF : inode

- Inode: attributs + localisation des blocs contenant les données du fichier
- Inode:
  - Id. du disque logique où est le fichier,
  - numéro du fichier sur ce disque
  - Type de fichier et droits d'accès
  - Nombre de liens physiques
  - Propriétaire, groupe propriétaire
  - Taille
  - Dates :
    - De dernier accès (y compris en lecture): atime
    - Date de dernière modification des données: mtime
    - Date de dernier modification de l'inode: ctime

Dossier/répertoire

- Deux grandes classes de fichiers :
  - Fichier ayant un contenu sur disque : fichiers réguliers, dossiers, liens symboliques, tubes
  - Ressources : Fichiers spéciaux (pilotes de périphériques, ràf, ...)
- Dossiers: listes de couples (nom, No inode)
- Un couple est appelé « lien physique » (hardlink)
- Du point de vue de l'utilisateur, un dossier contient des fichiers (fichiers réguliers, dossiers, ...)

Inodes/Nom: conséquences

- Créer/détruire un fichier: ajouter/retirer un couple dans le dossier
- opération nécessitant un droit au niveau du dossier pas du fichier
- Le système travaille avec des No d'inode, l'utilisateur avec les noms de fichiers :
  - les dossiers font le lien entre les deux :
    - On trouve le couple (nom, inode) du dossier où est le fichier
    - Pour trouver ce dossier, on applique le même principe (pour Unix, un dossier est aussi un fichier)

Fichiers: résumé:

- ce que l'utilisateur perçoit comme un fichier identifié par un nom peut se décomposer en trois notions sous unix :
  - un inode: informations (taille, dates, uid, gid, droits) et localisation des données sur disque
  - le contenu du fichier: les données qui y sont stockées
  - un lien physique: associe un nom à un inode. Un même inode peut avoir plusieurs lien.

Droits d'accès aux fichiers

- 3 types d'accès: lecture (R), écriture (W) et exécution (X)

Objet/Droit	R (lecture)	W (écriture)	X (exécuter)
fichier régulier	lire le contenu	modifier le fichier	exécuter le fichier
dossier	lister le contenu du dossier	modifier le contenu du dossier (y compris destruction de fichier)	utiliser le dossier dans un chemin ou s'y positionnner

- 3 classes d'utilisateurs: le propriétaire du fichier, le Groupe du propriétaire du fichier, les Autres utilisateurs.

type fichier	Propriétaire			Groupe du proprio			Autres utilisateurs		
-	R	W	X	R	-	X	R	-	X

- informations dans l'inode, affichage avec « ls », changement avec chmod, chgrp et chown

Droits d'accès : algorithme

- Si l'uid réel du processus est égal à l'uid du propriétaire du fichier: ce sont les droits du propriétaire qui détermine l'accès (y compris les refus d'accès)
- Sinon si le gid du processus est égal au gid du propriétaire du fichier: ce sont ces les droits du groupe propriétaires qui déterminent l'accès (y compris les refus d'accès)
- sinon, c'est l'entrée other qui est utilisée

Droits d'accès : algorithme

- Exemple (tordu) : soit un fichier f dont les permissions sont : R— RWX RWX et un utilisateur qui en est propriétaire et qui appartient au groupe propriétaire du fichier
  - L'utilisateur n'a que le droit de lecture alors qu'il appartient au groupe
  - Sous unix, les permissions ne sont pas cumulatives.
  - En résumé, on peut dire que sous unix, le particulier (utilisateur) l'emporte sur le général (groupe)

ACL posix: algorithme

- Si l'uid réel du processus est égal à l'uid du propriétaire du fichier ou à l'IUD d'une entrée utilisateur des ACL :ce sont les droits du propriétaire qui détermine l'accès (y compris les refus d'accès)
- Sinon si le gid du processus est égal au gid du propriétaire du fichier ou à l'IUD d'une entrée groupe des ACL : ce sont ces les droits du groupe propriétaires qui déterminent l'accès (y compris les refus d'accès)
- sinon, c'est l'entrée other qui est utilisée

ACL posix: conseils

- Les ACL POSIX vérifient le principe selon lequel le particulier l'emporte sur le général
- Utiliser des ACL utilisateur et des ACL groupe limite la lisibilité des ACL :
  - Pour savoir si un utilisateur a un droit, il faut vérifier si ce droit ne lui est pas refusé dans une ACL utilisateur et s'il existe une ACL utilisateur ou groupe ou other le lui donnant
- Conseil : n'utiliser que des ACL groupe (quitte à créer un groupe pour un seul utilisateur si nécessaire)

Droits d'accès (2): suid, sgid, sticky bit

- 3 autres « droits » spéciaux:
  - bit SUID: le programme s'exécute avec les droits de son propriétaire (au lieu de ceux de l'utilisateur qui le lance)
  - bit SGID: le programme s'exécute avec les droits du groupe propriétaires du fichier
  - sticky bit :
    - sur un fichier exécutable : (obsolète) maintient le fichier en mémoire après l'exécution pour diminuer le temps de la prochaine invocation
    - sur un dossier: seul le propriétaire du fichier a le droit de le supprimer. Exemple: /tmp/

Commandes de base: chmod

- chmod [-R] mode fichier ...
- -R: fichier est un dossier, chmod agit récursivement sur fichier et sur son contenu
- mode:
  - forme numérique: 644
    - pour u: 400 (r), 200 (w) et 100 (x)
    - pour g: 40 (r), 20 (w) et 10 (x)
    - pour o: 4 (r), 2 (w) et 1 (x)
  - forme symbolique: [ugo][+-=][rwxXstguo]

chmod: exemples

commande de base: umask

- définit les droits d'accès par défaut d'un fichier
- les droits sont le complément du paramètre d'umask: on laisse tout sauf les droits précisés
- Exemple:
  - umask 002 : mode par défaut: RWXRWXR-X (tout sauf 002)
  - umask 026: mode par défaut: RWXR-X--X (tout sauf 026)
  - umask a=rx,gu+w: mode par défaut: RWXRWXR-X
  - umask -S : affiche le l'état courant sous forme symbolique : u=rwx,g=rwx,o=rw dans notre exemple.

Commandes de base: chown, chgrp

- chown -R [-H | -L | -P] proprio[:groupe] fichier
- chgrp -R [-H | -L | -P] groupe fichier ...

chown/chgrp: exemples

Commandes de base: ls

Commandes de base: cat

Commande de base: stat

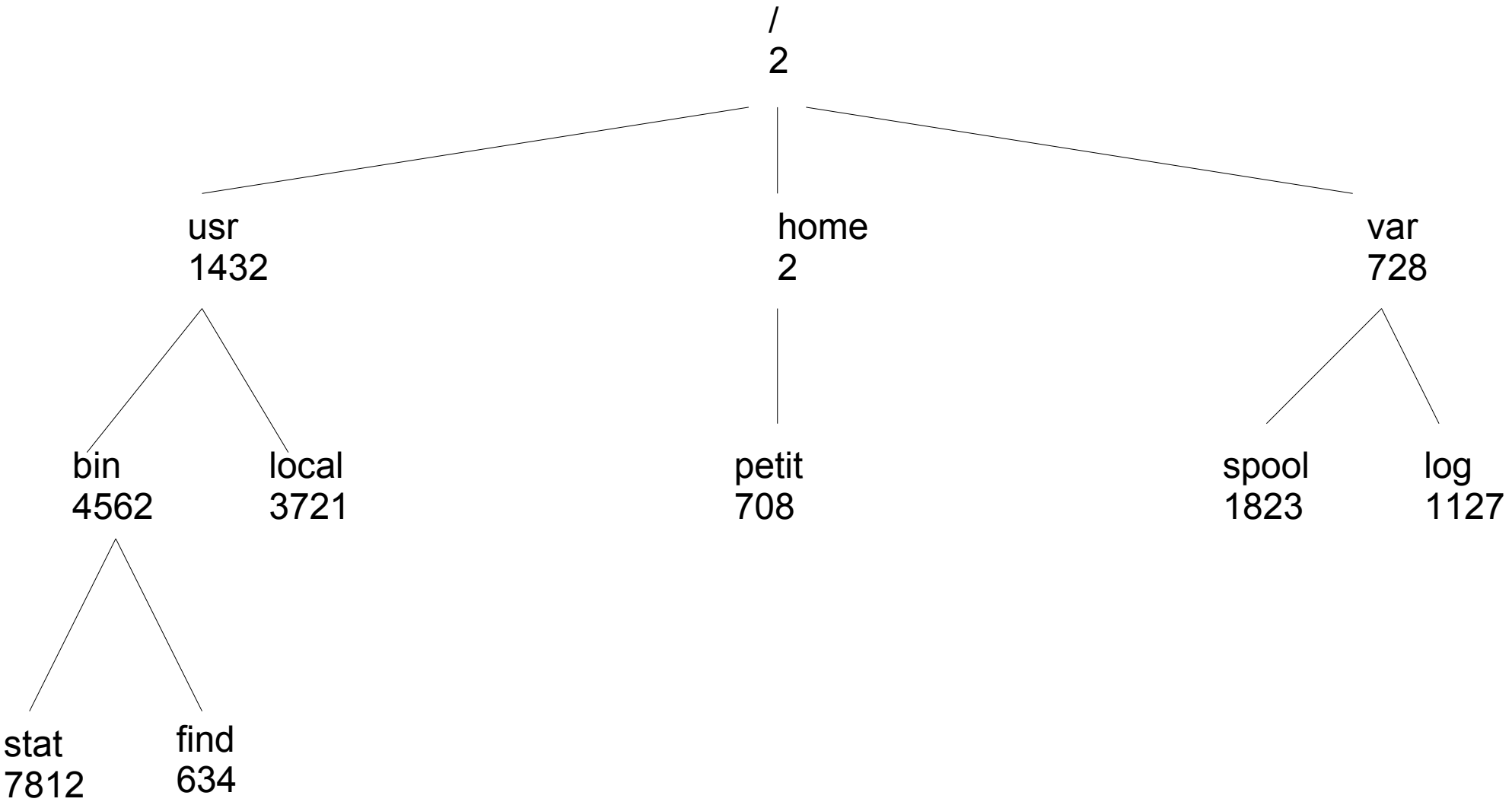
Exemples

- Stat fichier (noter ctime, mtime et atime)
- Cat fichier
- Stat fichier (atime a changé)
- Chmod fichier
- Stat fichier (ctime a changé)
- Modif fichier
- Stat fichier (mtime a changé)

Arborescence

- Sous unix, on a une arborescence unique (donc pas de C:\, D:\, ...comme sous windows)
- Le disque système contient la racine absolue /
- toute l'arborescence est sous cette racine absolue
- Les systèmes de fichiers des autres partitions s'intègrent dans l'arborescence en prenant la place d'un dossier existant
- la racine d'un système de fichier a 2 comme numéro d'inode

arborescence



monter un système de fichier

- commande mount/umount
- /etc/fstab
  - des associations système de fichier/point de montage
  - notamment: les partitions à monter au démarrage du système
  - l'ordre des lignes est important pour mount, umount et fsck
  - syntaxe: periph pointDeMontage typeSGF options fsfreq fspassNo
  - demo: exemple de fstab
- fsck, df, du

algo de recherche

- /usr/bin/stat
- algo de localication:
  - examiner le contenu du dossier d'inode 2 pour trouver le No d'inode du dossier usr : 1432 par exemple.
  - examiner le contenu de dossier d'inode 1432 pour trouver le No d'inode du dossier bin. 4562 par exemple
  - examiner le contenu de dossier d'inode 4562 pour trouver le No d'inode du fichier stat. 7812 par exemple
  - exécuter le fichier d'inode 7812

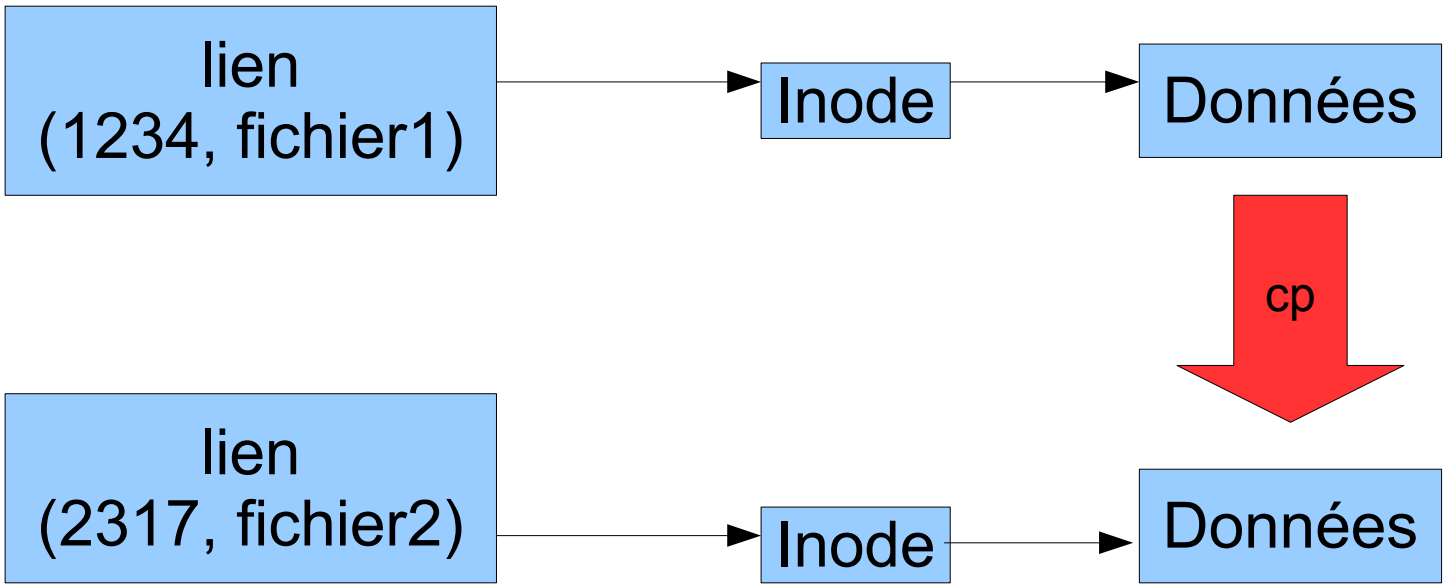
Chemin

- /usr/bin/stat: chemin absolu du fichier stat
- chemin absolu: chemin depuis la racine absolue
- notion de dossier courant
- chemin relatif: chemin depuis le dossier courant

Commandes de base:

- pwd : indique le dossier courant
- cd : changer de dossier courant
- mkdir: pour créer un dossier
- rmdir: détruit les dossiers vides

commande de base: cp



cp fichier1 fichier2

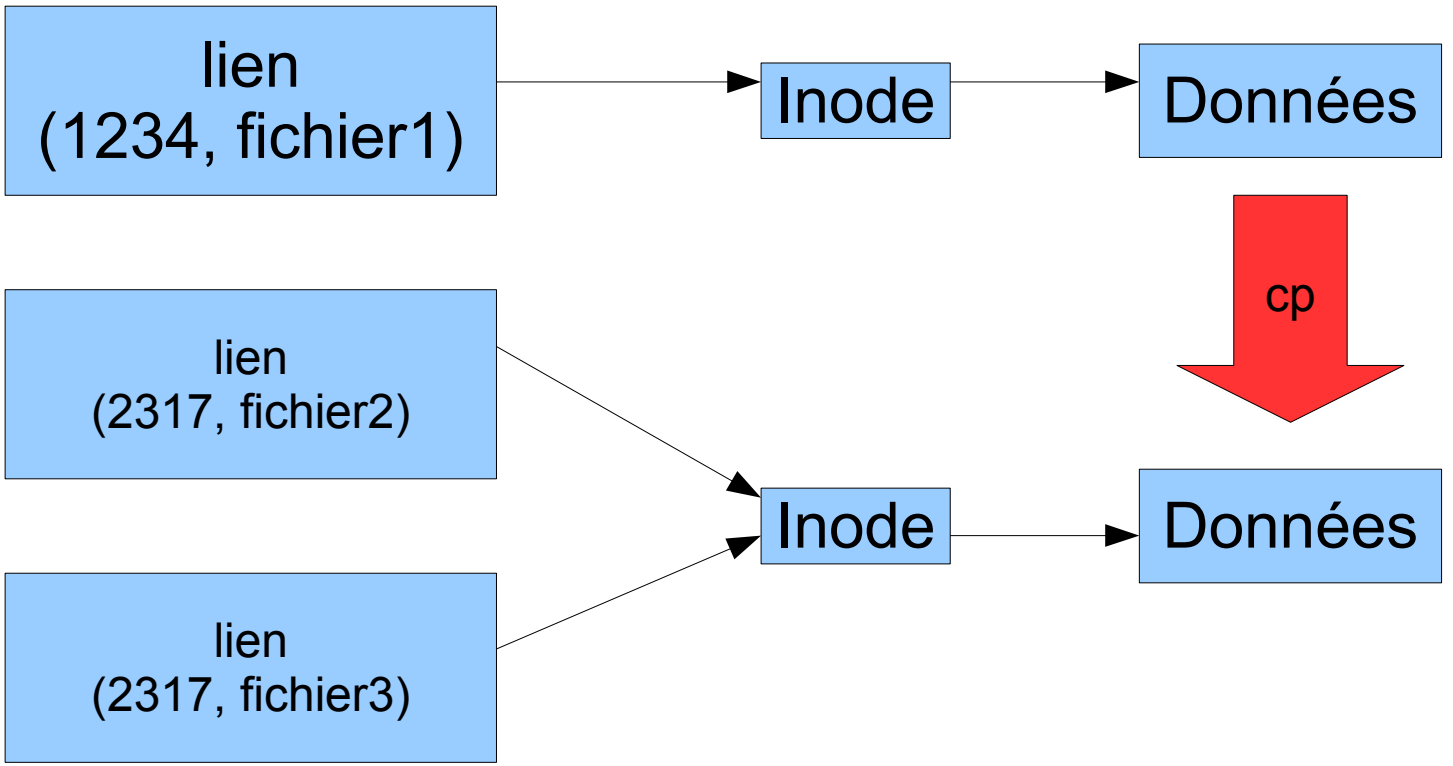
Commandes de base: cp

- copie des données d'un fichier (source) dans un autre (cible) :
  - la cible n'existe pas : création d'un nouvel inode et recopie des données du fichier
  - la cible existe: inode destination inchangée, recopie des données du fichier dans la cible

cp (2)

- gnu cp: en standard sous Linux, installable facilement ailleurs
- fournit des options non standard mais pratiques
- ràf

cp et liens physiques



cp fichier1 fichier2

Commandes de base: rm

- suppression d'un lien d'un fichier ou plusieurs fichiers: `rm [-fiRr] fichier1 ...`
- options:
  - -i: demande de confirmation pour tout fichier à supprimer (aff sur stderr et lecture sur stdin)
  - -f: supprime les messages d'erreur lorsqu'un fichier n'existe pas et supprime la demande d'acquittement si l'utilisateur de rm n'a pas les droits d'écriture sur le fichier à supprimer
  - -R ou -r: supprime récursivement le contenu d'un dossier avant d'appliquer rmdir au dossier.

- rm :exemples

Commandes de base: mv

- mv [-fi] source destination
  - renomme un lien. Source et fichier sont des fichiers réguliers
- mv [-fi] source1 ... destination
  - renomme les sources en les déplaçant **dans** le dossier destination
- la commande fonctionne aussi si sources et destinations sont dans des systèmes de fichiers différents.
- la seconde forme est utilisée si la destination est un dossier existant

mv: exemples

- mv [-fi] source destination
  - renomme un lien. Source et fichier sont des fichiers réguliers
- mv [-fi] source1 ... destination
  - renomme les sources en les déplaçant **dans** le dossier destination
- la commande fonctionne aussi si sources et destinations sont dans des systèmes de fichiers différents.
- la seconde forme est utilisée si la destination est un dossier existant

Commandes de base: ln

- ln fichier nouveau_lien_physique
- ln -s fichier lien_symbolique
- options:
  - -s: crée un lien symbolique
  - -f: force la création même si la destination existe déjà
  - --: fin des options (pour permettre le traitement d'un fichier dont le nom commence par « « - »