Généralités

- rôles d'un système d'exploitation (OS ou opérating System)
  - machine virtuelle qui cache la machine physique
  - gestion des ressources matérielles (mémoire, périphériques, ...)
- Unix
  - multi tâches et multi utilisateurs
  - axiome: « tout est fichier »
  - interface standard: POSIX
  - unix: plusieurs famille d'OS : BSD, SYS V, Linux, ...
    - cf http://fr.wikipedia.org/wiki/Tableau_synoptique_des_syst%C3%A8mes_d'exploitation

Rôle d'un administrateur système

- ajouts et suppressions d'utilisateurs
- ajout, suppression, configuration de matériel
- sauvegarde et restauration
- installation, mise à jour de logiciels
- surveillance du système:
  - sécurité
  - monitoring
- documentation locale
- rédaction de fiches de procédures, cahier des charges, ...
- aide aux utilisateurs

Principes de base de l'administration

- tout système doit avoir un administrateur
- complexité :
  - de plus en plus de machines
  - des systèmes hétérogènes
- méthodes de travail
  - rigueur (doc, gestion de version, validation, ...)
  - automatiser les tâches qui peuvent l'être
  - documentation: fiches de procédure, cahier des charges, ...
  - pas de travail de fond: le vendredi soir, après un pot :-)

Droits d'accès aux fichiers et aux processus

- fichiers: possède un propriétaire (UID) et un groupe propriétaire (GID)
- le propriétaire seul à pouvoir modifier les permissions du fichiers
- processus:
  - UID et GID réel: ceux de l'utilisateur du programme (et pas ceux du propriétaire du fichier sur disque)
  - UID et GID effectifs: pour déterminer les droits d'accès
- bits SETUID ou SETGIUD

Root

- root (superutilisateur): tout compte d'UID 0
- peut exécuter toute opération **valide** sur n'importe quel fichier ou processus
- devenir root
  - connexion directe en tant que root: déconseillé, notamment via réseau
  - su : changement d'identité
  - sudo: exécuter certaines commande en tant que root
- il est important de journaliser les actions effectuées en tant que root (sudo, snoopylogger, ...)

les autres pseudo-utilisateurs

- daemon : propriétaire des logiciels systèmes (uid 1)
- nobody: utilisateur sans droit
  - utilisé par nfs notamment
  - utilisé par certains daemon
  - ne doit possèder aucun fichier
- squid, bind, ...: pratique actuelle

méthodes d'administration

- installation de logiciels
  - des packages binaires via l'un des outils de gestion de packages du système d'exploitation :
  - des logiciels livrés en source à recompiler
  - des logiciels non livrés avec le système d'exploitation
- configuration:
  - via l'édition de fichier de configuration (=> il faut maîtriser un outil d'édition de texte par plateforme)
  - via des commandes d'administration
  - utilisation d'outils intégrés (smit (AIX), sam (HP UX), webmin, admintool (solaris)
  - via des scripts ou des extension d'outil (webmin) maison

Documentation

- man: les pages de manuel d'unix, sections du manuel
  - man commande
  - exemples :
    - man 1 kill: kill commande utilisateur (section 1 du man)
    - man 2 kill: kill appel système (section 2 du man)
    - apropos ou man -k
- documentation du système d'exploitation, /usr/doc, /usr/share/doc, ...
- WeB (google est votre ami notamment pour les messages d'erreurs)
- groupe de news USENET, forums WeB
  - fr.comp.os.* par exemple

le démarrage des PC

- MBR:
  - premier bloc de données du periph d'amorçage
  - 446 octets pour le gestionnaire d'amorçage
  - 64 octets pour la table des partitions (16 par partitions)
  - 2 octets à AA55 (valeur fixe)
- PBR: partition boot loader: en début de partition, chargé par le chargeur d'amorçage du MBR ou un autre chargeur d'amorçage
- MBR microsoft : charge le pbr de la première partition principale active
- gestionnaire d'amorçage: lilo, grub & Co

démarrage du système

- chargement et initialisation du noyau
- détection des périphériques
- création des processus systèmes, montage de la partition racine /en lecture seule, création du processus init
- intervention de l'opérateur (en cas de démarrage en mode monoutilisateur)
- exécution des scripts de démarrage
- passage en mode multiutilisateur

init

- cf man telinit
- Rôle de init
  - exécution ds scripts d'initialisation
  - gestion des terminaux
  - ancêtre de tous les processus
- 2 types de démarrage :
  - BSD
  - SYSV

mode monoutilisateur

- init lance un processus qui lance un shell root
- le processus de démarrage reprend après
- ce modecorrespond aux runlevel 1 ou S (le sens dépend de la distribution de l'OS)
  - S: réparation des systèmes de fichiers ou ~: pas de daemon, racine en lecture seule est la seule partition montée
    - il faut la remonter en lecture/écriture et monter les autres partitions si nécessaire
  - 1: administration, certains daemon sont actifs, les partitions sont toutes montées (Sys V, Suse, ...ràf: à préciser sous debian)

états du système

- BSD: monoutilisateur ou multiutilisateur
- Sys V, Linux:
  - plusieurs états (Run Level)
  - sens (dépend du système, voire de la distribution) :
    - 0: arrêt de la machine (halt)
    - 1: monoutilisateur (administration)
    - 2 à 5 : multiutilisateur
    - 6: redémarrage (reboot)
    - s ou S: monoutilisateur (maintenance)
  - le « run level » par défaut est défini dans /etc/inittab
  - connaître son « Run level »: who -r, runlevel (linux)
  - changer de « run level »: telinit No

scripts de démarrage BSD

- varient selon le système BSD
- l'idée de base: deux scripts (/etc/rc et /etc/rc.local) pilotent le démarrages du sytème
- Scripts de démarrage FreeBSD et autres BSD libres (ràf: dans une version ultérieure de ce document)

scripts de démarrage Sys V et Linux

- le système le plus répandu
- /etc/inittab:
  - indique à init ce qu'il faut faire pour chaque niveau d'exécution
  - au démarrage: init passe par tous les niveaux entre 0 et le niveau d'exécution par défaut;
  - idem dans l'autre sens à l'arrêt
  - les scripts de démarrages situées dans /etc/rcN.d sont démarrés via inittab
  - en pratique: de nos jours, on ne modifie que rarement inittab, on s'appuie sur les scripts (cf ci-dessus)

inittab

- Demo:
  - syntaxe du fichier
  - man inittab

Scripts de démarrage Sys V (suite)

- /etc/init.d ou /sbin/init.d (HP-UX):
  - contient la copie d'origine des scripts de démarrage
  - un script par daemon ou aspect du système
  - les scripts comprennent les arguments start et stop (et parfois d'autres arguments comme restart)
- /etc/rcN.d (N=0, 1, 2, ... : run level)
  - contient un lien symbolique vers le scripts d'origine. Le nom du lien peut avoir deux formes:
    - KXX: le service doit être arreté dans ce run level
    - SXX: le service doit être démarré dans ce « run level »
  - outils de gestion des scripts de démarrage : update-rc.d (linux debian)

Scripts de démarrage Sys V (suite)

- processus d'appel des scripts par init
  - recherche de /etc/rcN.d
  - si init augmente son niveau d'exécution: execution des scripts commençant par S dans l'ordre des No
  - si init diminue son niveau d'exécution: execution des scripts commençant par K dans l'ordre des No
- les scripts peuvent être utilisés pour arrêter ou redémarrer manuelle un service
  - exemple: /etc/inid.d/networking restart
- DEMO: exemple d'ajout de script avec update-rc.d

Scripts de démarrage: la relève

- Scripts SYS-V : supposent un système complètement opérationnel au boot : difficilement compatible avec les périphériques amovibles
- Gestion minimale des dépendances entre actions :
  - exemple: le montage d'une entrée de /etc/fstab peut nécessiter un outil présent dans /usr qui peut être monté via réseau. Il faut donc attendre que le réseau soit opérationnel
- la configuration des processus à lancer est dispersée :
  - dans /etc/init.d & Co, dans la crontab, dans

Scripts de démarrage: initng

- initing : « init new generation »
- permet le lancement asynchrone des scripts
- gestion des dépendances entre scripts
  - un script est lancé quand tous les scripts dont il dépend ont été lancés
- Exemple :

```
service system/my_service {
  need = system/initial net/all;

  exec start = /sbin/my_service --start –option;
  exec stop = /sbin/my_service --stop --option;
}
```

Scripts de démarrage: Sun Solaris Service Management Facility (SMF)

- dans une version ultérieure de ce document
- cf http://www.sun.com/bigadmin/content/selfheal/

Scripts de démarrage: upstart

- Un outil unique qui s'appuie sur des événements : les scripts sont lancés lors de certaines évènements
  - le système a démarré;
  - un système de fichier (SGF) a été monté
  - le SGF racine est en lecture/écriture
  - un périphérique block a été ajouté au système
  - il est une certaine heure (remplace cron)
  - un processus vient d'être lancé ou s'est arrêté
  - un fichier a été modifié
  - un périphérique réseau a été détecté
  - une connexion réseau a lieu (remplace inetd)
  - ...

Scripts de démarrage: upstart

- The two states shown in red (“waiting” and “running”) are rest states, normally we expect the job to remain in these states until an event comes in, at which point we need to take actual to get the job into the next state.
- The other states are temporary states; these allow a job to run shell script to prepare for the job itself to be run (“starting”) and clean up afterwards (“stopping”). For services that should be respawned if they terminate before an event that stops them is received, they may run shell script before the process is started again (“respawning”).
- Jobs leave a state because the process associated with them terminates (or gets killed) and move to the next appropriate state, **following the green arrow if the job is to be started or the red arrow if it is to be stopped**. When a script returns a non-zero exit status, or is killed, the job will always be stoped. When the main process terminates and the job should not be respawned, the job will also always be stopped.
- As already covered, events generated by the init daemon or received from other processes cause jobs to be started or stopped; also manual requests to start or stop a job may be received.
- The communication between the init daemon and other processes is bi-directional, so the status of jobs may be queries and even changes of state to all jobs be

Scripts de démarrage: upstart

- Exemple (version simplifiée du script de démarrage des scripts d'init SYSV de niv. 2 sur ubuntu 6.10)

```
start on startup
stop on shutdown
stop on runlevel-3

script
  set $(runlevel --set 2 || true)
  exec /etc/init.d/rc 2
end script
```

Scripts de démarrage: lauchd

- lauchd est fourni avec MacOS X
- similaire à upstart
  - des scripts sont lancés quand certains évènements se produisent
- différence avec upstart: les évènements possibles sont très limités:
  - démarrage du système
  - fichier modifié ou placé dans une file d'attente
  - date/heure atteinte (remplace cron)
  - connexion sur un port donné (remplace inetd)

Scripts de démarrage: la relève

- upstart (ubuntu) :
  - http://www.netsplit.com/blog/articles/2006/08/26/upstart-in-universe
  - http://upstart.ubuntu.com/
- launchd (MacOS-X) : http://developer.apple.com/macosx/launchd.html
- Sun Service Management Facility : http://www.sun.com/bigadmin/content/selfheal/
- synthese: http://www-128.ibm.com/developerworks/library/l-boot-faster/index.html?ca=dgr-lnxw09Fas

arrêt du système

- init est à la source de tout
  - arrêt du système: demander à init de tuer proprement ses processus fils
  - shutdown [-r|-h] heure message : pour arrêter ou rebooter le système à l'heure indiqué
  - reboot: équivaut à shutdown -r ou à telinit 6
  - halt: équivaut à shutdown -h ou à telinit 0

processus d'arrêt du système

- écriture de l'évènement dans les journaux
- exécute les scripts KXX (init SYSV)
- init tue ses processus fils
- vide les tampons disque (sync)
- démonte les systèmes de fichiers quand les écritures sont terminées
- arrête la machine

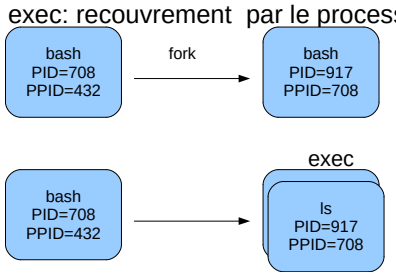
cycle de vie d'un processus

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

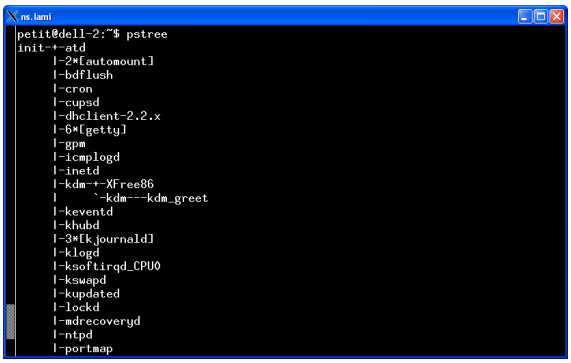
- un processus (processus fils) est toujours créé par un autre processus (processus père):
  - fork: création d'une copie du processus père
  - exec: recouvrement par le processus fils



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



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

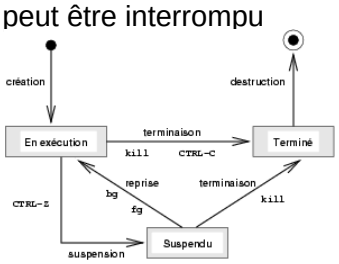
commande ps

- 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

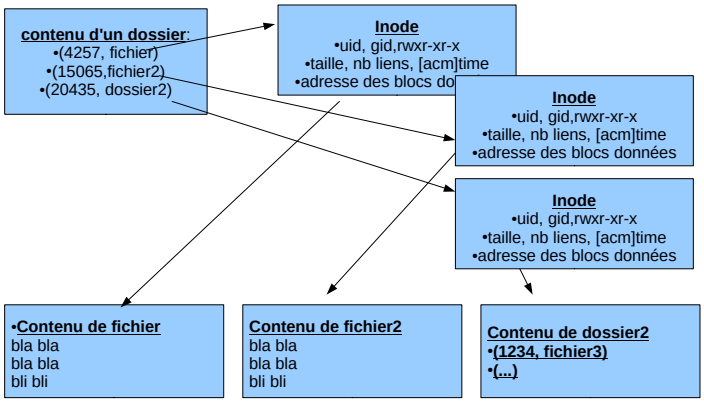
Systèmes de gestion de fichiers (SGF)

- 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



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

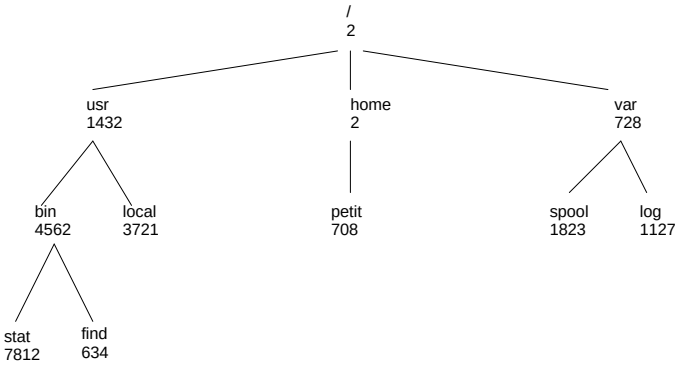
Exemples

- 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

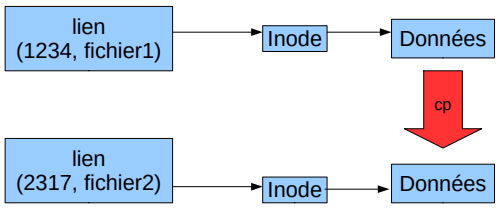
Chemin

- /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

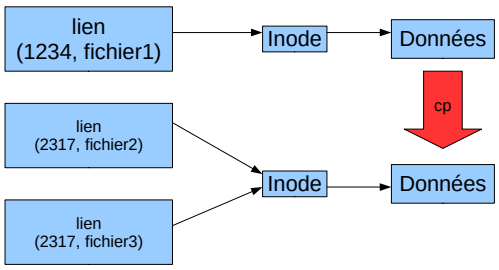
Commandes de base: cp

- 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 « « - »

Exemples

Sauvegarde versus Archivage: sauvegarde

- sauvegarde: c'est dupliquer et externaliser les données pour se protéger :
  - des erreurs humaines
  - des crash matériel
  - des erreurs logicielles
  - des sinistres (incendie, inondation, ...)
  - de la malveillance (vol, virus, ...)
- la durée de vie d'un jeu de sauvegarde est limitée dans le temps (quelques mois au plus)
- Sauvegarde des données / du système

Sauvegarde versus Archivage: sauvegarde

- on sauvegarde certaines données pour se protéger de certains risques
  - faire la liste des risques dont il faut se protéger
  - adapter le cahier des charges des sauvegardes et des autres mesures en fonction des ces risques
  - les sauvegardes ne sont qu'un des éléments permettant de garantir la continuité du service à côté d'autres : disques raid, redondance des serveurs, ...

Sauvegarde versus Archivage: archivage

- l'archivage, c'est le stockage hors ligne de données peu utilisées
  - le temps d'accès a souvent peu d'importance
  - on peut avoir une hiérarchie de mode de stockage (du plus rapide au moins rapide (bande à aller chercher) suivant la fréquence de l'utilisation des données concernées)
- la durée de vie d'une archive se compte au minimum en années, voire en décennies
- il faut donc penser à l'obsolescence des matériels, des supports, des logiciels, des standards utilisés

Solutions qui ont prouvé leur inefficacité

- Faire faire les sauvegardes par les utilisateurs : pour dégager sa responsabilité mais aucune garantie qu'elles seront faites
- Sauvegardes sur des supports peu fiables (disquettes, DAT, ...)
- Sauvegarde en écriture seule : il faut valider sauvegarde et procédures de restauration
- Sauvegarde d'un système en cours d'exécution
- Un seul support de sauvegarde
- Sauver sur une partition du même disque
- Pas de sauvegarde hors site (incendie, ...)

Sauvegardes : procédure/planification

- Planifier les sauvegardes, tester leur réalisation
  - il faut avoir l'assurance que les sauvegarde prévues ont eu lieu
  - les procédures de sauvegardes doit être écrites
  - procédures testées
  - procédures et planification validées par les utilisateurs/propriétaires des données

Sauvegardes : planification

- choix des données à sauvegarder :
  - données des utilisateurs (y compris : boîtes aux lettres, profil, ...), fichiers de configuration, ..;
    - retrouver un système en état suppose de réinstaller le système puis de restaurer les données
    - volumétrie plus faible
  - Système entier avec procédure de redémarrage:
    - restauration directe et rapide d'un système opérationnel
    - volumétrie plus importante
- périodicité, choix des données ont un impact fort sur la volumétrie et donc sur le coût =>compromis

Problèmes liés à la volumétrie:

- Coût
- Charge réseau
- Durée des sauvegardes
- Indisponibilité des serveurs/applications dans le cas où le logiciel de sauvegarde impose l'arrêt des logiciels.

Restauration

- procédures de restauration
  - écrites
  - au résultat validé par les propriétaires des données (pour ne pas en oublier)
  - **testées régulièrement en grandeur réelle** de façon à garantir :
    - que toutes les données pertinentes ont été sauvegardées
    - d'être capable de tout restaurer correctement

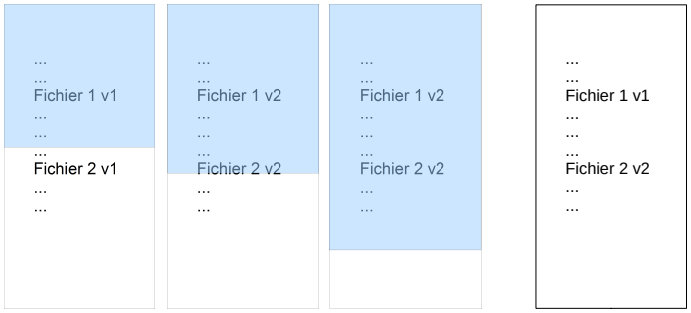
Fiablisation

- utiliser des média fiables
  - éviter disquettes, CD RW, DAT, ..et leur préférer CD R (bof), DLT, AIT, ...
  - varier les marques de media pour palier un défaut de fabrication dans une série, ...)
- gérer le vieillissement des média de sauvegarde/archivage (reprise sur des média récents, ...)

Fiabilisation (2)

- fiabiliser l'environnement des media de sauvegarde:
  - inondation, incendie, vol (coffre ignifugé, ...)
  - sauvegarde hors site (penser à la confidentialité des données, au risque de vol, à l'interception des données, ...)
- indexer les données pour s'y retrouver dans le volume total des jeux de sauvegardes
  - indexer les données
  - étiqueter les media

Sauvegarde live: problématique



Si l'application modifie plusieurs fichiers durant la sauvegarde, les versions sauvées peuvent ne pas être cohérentes.

Sauvegarde live: solutions

- Arrêt de l'application pendant la sauvegarde:
  - garantit de plus qu'aucun verrou n'empêchera l'accès à un fichier
- Instantané (Snapshot): image des blocs du sgf, en cas d'effacement de fichier, les blocs ne sont pas réalloués tant que la sauvegarde n'est pas finie
  - Suppose un support dans le sgf
  - Proposé par les NAS, par afs, LVM, UFS2 (FreeBSD, ...), ...
  - Application supplémentaire : proposer aux utilisateurs une image des états antérieurs de leur compte à moindre coût.

Sauvegarde live: snapshot

- Là, on mettra une illustration du fonctionnement des snapshot
- Fait en live au tableau.

Sauvegardes incrémentales/différentielles

- Sauvegarde incrémentales : fichiers créés ou modifiés depuis la sauvegarde précédente
  - Diminue le volume à sauver
  - Restauration nécessite toutes les sauvegardes, restaure toutes les versions d'un même fichier
  - Peu adapté si la totalité des fichiers changent
- Sauvegarde différentielle : fichiers créés/modifiés depuis la dernière sauvegarde de référence
  - Diminue le volume à sauver
  - Plus de volume qu'en incrémental
  - Restauration nécessite la sauvegarde de référence et la dernière sauvegarde différentielle

Exemple:

- Comparer la taille des sauvegardes et les procédures de restauration dans les cas suivants : (ST: sauvegarde totale, I: sauvegarde incrémentale, D: sauvegarde Différentielle)
- Cas 1) ST, I1, I2, I3, I4, I5, I5
- Cas 2) ST, D1, D2, D3, D4, D5
- Cas 3) ST, I1, I2, I3, D1, I4, I5

Dump: un outil de sauvegarde sous unix

- Dump est un outil unix
- Il est efficace (travail directement au niveau du sgf)
- Sauvegarde non portable d'un unix à un autre (lié au sgf)
- Niveau (0 à 9) permettant un gestion très souples des sauvegarde incrémentales/differentielles:
  - Une sauvegarde niveau *n* sauvegarde tous les fichiers modifiés depuis la dernière sauvegarde de niveau *n-1*