Séance No 1

- Définition
- Historique
- Boucle d'interprétation, commandes simples: forme générale
- Erreurs classiques et aide en ligne
- Systèmes de gestion de fichiers (SGF)
- Fichiers, inodes, dossiers, arborescence
- droit d'accès
- commandes sur les fichiers/dossiers

Définition

- SHELL: programme en mode texte assurant l'interface entre l'utilisateur et le système unix.
- S'utilise :
  - En interactif depuis une fenêtre terminal (xterm, connexion distante texte, ...) : interpréteur de commande
  - Pour réaliser des scripts (fichiers de commandes) : langage de programmation

Shell: où que je clique ?

- On ne clique pas : ça s'utilise avec une souris à 105 touches et sans boule : un clavier :-)
- L'accès est moins immédiat que celui d'une interface graphique
- Plus de liberté/possibilités qu'avec une interface graphique
- Langage de programmation: possibilité d'exprimer des requètes complexes
- Utilisation interactive ou pour écrire des fichiers de commandes (scripts)

Historique

- Les deux shells des origines sont à l'origine de deux familles de shells aux syntaxes incompatibles :
  - Le shell le plus ancien : sh ou Bourne shell écrit dans les années 70 par Steve Bourne. Tout système système unix a un shell /bin/sh qui est un bourne shell (ou un shell compatible);
  - Le csh: écrit à la même époque par Bill Joy incompatible avec le bourne shell mais offrant quelques fonctionnalités supplémentaires (historique des commandes, aliases, contrôle de tâches, ...

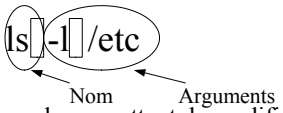
Historique (2)

- Ksh: korn shell (David Korn, 1983) sur la base du bourne sh. Le ksh 88 (ou +) est livré avec tous les unix commerciaux. Base de la norme IEEE Posix 1003.2;
- Tcsh: un shell évolué de la famille csh utilisé dans les années 90 comme shell interactif;
- Bash: Bourne Again sh, le shell de la FSF. Compatible posix 1003.2. Le shell de base des distribution linux.
- Zsh:
- De nos jour, il est conseillé d'utiliser un shell compatible posix: ksh, bash et zsh.

Boucle d'interprétation

- Le shell est un programme qui réalise la boucle suivante :
  - Boucle :
    - Lire la ligne de commande
    - Décoder la ligne de commande
    - Exécution de la ligne de commande en créant un processus dans le cas de commandes externes
    - Attendre la fin de l'exécution du processus
    - Retourner en début de boucle

commandes simples: forme générale



- arguments:
  - paramètres optionnels permettant de modifier le comportement de la commande
  - liste des entités auxquelles doit s'appliquer la commande (nom de fichier, processus, utilisateur, ...)
- Exemples:
  - mozilla
  - mozilla -P toto www.univ-evry.fr
  - ls -lrt /etc
  - find ~ -name *.avi -exec rm -f {} \;

quelques commandes simples

- who: liste des utilisateurs ayant une session en cours sur l'ordinateur
- w: idem mais indique aussi ce qu'ils font
- date: date courante
- echo: affiche ses arguments séparés par une espace

Exemples

- Le shell va servir à lancer des commandes internes ou externes
- Exemple de session :

```
#un commentaire commence par #
# lister les fichiers présents
# dans le dossier /etc
ls -l /etc
# liste des utilisateurs connectés
# sur l'ordinateur
who
```

Erreurs

- 5 causes classiques d'erreurs

1) syntaxe ou chemin incorrect (commande inconnue, ...)

2)paramètres incorrects (fichier inconnu, ..)

3)droits d'accès : permission refusée (accès à un fichier, ...)

4)options invalides (syntaxe des options de la commande)

5)erreur de conception : le comportement n'est pas celui attendu

•A l'aide

- le manuel
- option « --help » de certaines commandes
- la documentation de votre système d'exploitation ou du programme posant problème
  - souvent /usr/share/doc, /usr/local/share/doc
- recherche sur le WeB: quelqu'un d'autre a forcément déjà eu ce problème

•Le manuel

- dans une version ultérieure de ce document
- les sections du manuel: ràf
- exemples d'utilisation: ràf + exemple dans 2 sections

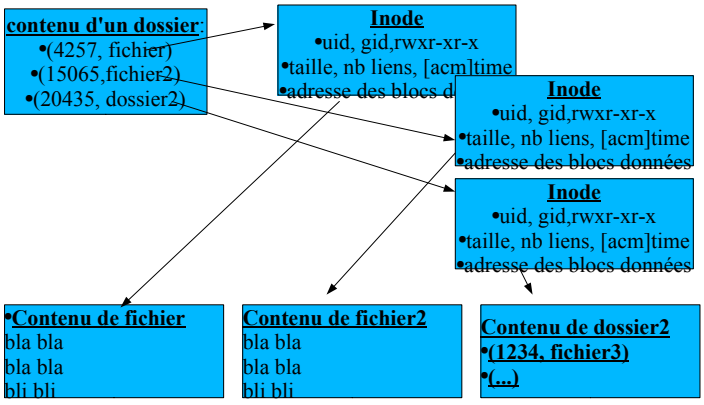
Systèmes de gestion de fichiers (SGF)

- 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
- 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 :
  - Ce sont les dossiers qui permettent de faire le lien entre les deux
  - On trouve le coupe (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 (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

- chmod ràf

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

-

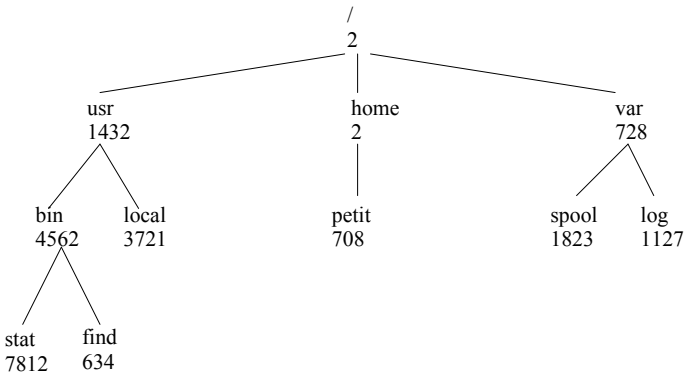
Exemples

- 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



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

Commandes de base: cd

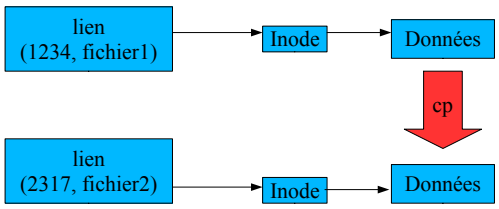
- sans argument: le dossier courant devient le dossier personnel
- cd *dossier* : le dossier courant devient *dossier*
- . désigne le dossier courant
- .. désigne le dossier père du dossier courant
- ~ désigne le dossier personnel de l'utilisateur. 2 commande équivalentes :
  - cd
  - cd ~

Commandes de base: mkdir

Commandes de base: rmdir

- détruit les dossiers vides

commande de base: cp



cp fichier1 fichier2

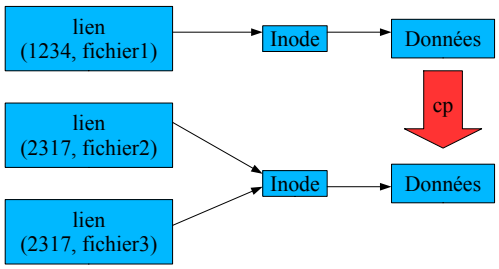
Commandes de base: cp

- 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 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 source destination
  - renomme un lien. source et fichier sont des fichiers régulier
- mv source1 ... destination
  - renomme les sources en les déplaçant dans le dossier destination

Commandes de base: ln

- ln fichier nouveau_lien_physique
- ln -s fichier lien_symbolique

Exemples

Bilan

- A la fin de cette première séance, vous devez :
  - connaître les notions de systeme de gestion de fichiers, fichiers, inode, dossier, chemin
  - connaître la forme générale d'une commande
  - savoir utiliser le manuel unix
  - savoir vous déplacer dans une arborescence,
  - créer/déplacer/copier des fichiers, des dossiers