Introduction à Unix/Linux: Cours 2.

Arborescence et Shell

Michaël Quisquater (Maître de Conférences,UVSQ)

Première partie I

Premier contact avec Unix/Linux

Avertissement

- Les systèmes d'exploitation Linux suivent presque la norme Posix
- Toutes les distributions Linux ne sont pas identiques (Redhat, Debian, Ubuntu, Fedora, etc)
- Ces distributions diffèrent par les choix des applications mais partagent le même noyau (le noyau Linux qui possède différentes versions)
- Lorsque l'on connaît le fonctionnement d'une distribution Linux il n'est pas trop difficile de passer à une autre
- Le présent cours est basé sur la distribution Debian wheezy 7.6 (commande `lsb_release -a`)

Système de fichier : Quésako ?

- Au sein d'une partition, les données (programmes, ...) ne sont pas stockées n'importe comment : il serait très difficile de les retrouver par la suite ! !
- Un système de fichiers est une façon d'ordonner les données sur un disque. Il existe plusieurs méthodes pour cela :
  1. Allocation contiguë
  2. Allocation au moyen d'une liste chaînée
  3. Allocation au moyen d'une liste chaînée indexée
  4. Les noeuds d'information

Allocation contiguë

- Le principe est similaire à une table des matières dans un livre
- Il suffit de constituer une table contenant l'adresse mémoire (disque) du premier bloc de chaque fichier ainsi que la taille de chacun d'entre eux.
- - Avantage : L'accès au fichier est rapide (une seule opération)
  - Inconvénients :
    - Il faut connaître la taille du fichier à l'avance avant écriture : quid si modification ?
    - Cela fragmente le disque si on retire des fichiers et qu'on essaie de combler les trous ensuite

Allocation au moyen d'une liste chaînée

- Les fichiers sont stockés sous forme de listes chaînées de blocs.
- Comme pour l'allocation contiguë, on constitue une table contenant l'adresse mémoire (disque) du premier bloc de chaque liste chaînée correspondant à un fichier.
- - Avantage : Tous les blocs sont utilisés. Il n'y a donc plus de fragmentation du disque.
  - Inconvénients :
    - L'accès est très lent.
    - Une partie de chaque bloc est utilisée pour le chaînage. La taille de l'espace réservé pour le fichier n'est donc plus une puissance de 2 ce qui est un problème étant donné le format de lecture et d'écriture des fichiers.

Allocation au moyen d'une liste chaînée indexée

- Les inconvénients des listes chaînées peuvent être évités en sauvegardant les blocs des fichiers dans une table.

bloc physique	0	1	2	3	4	5	6	7	8	9
Début de fichier	6	5		7		9		0		4
		A		B						

  - Blocs du fichier A : 1,5,9 et 4.
  - Blocs du fichier B : 3,7, 0 et 6

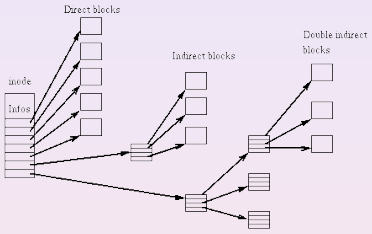
Allocation au moyen d'une liste chaînée indexée (suite)

- Avantages :
  - L'intégralité de la taille des blocs est disponible pour les données.
  - L'accès aux blocs est rapide
- Inconvénients :
  - La taille de la table pour sauvergarder les blocs des fichiers est assez grande et doit être gardée en mémoire. p.ex 1.5Mo pour un disque de 0.5 Go.

Remarque : Les systèmes de fichiers FAT (File Allocation Table) sont de ce type (MS-DOS, Windows jusqu'à Millenium). Ils sont précédés d'un Volume Boot Recorder qui est similaire au MBR.

Les noeuds d'information

- Cette méthode consiste à associer à chaque fichier une petite table appelée noeud d'information (i-node).
- Chaque i-node contient des attributs du fichier (propriétaire, type, permission, date de modif etc) et une table d'adresses disques.
- Les premières adresses disques pointent directement sur des blocs du disque.



- Les 3 adresses suivantes vers un bloc d'indirection simple, double et triple respectivement.
- Les systèmes de fichiers NTFS (Windows NT jusqu'à 7) et ext2, ext3 et ext4 (Linux) utilise cette méthode.

Les noeuds d'information : cas de Linux

Dans le cas de Linux, le système de fichiers se compose :

- Un Volume Boot Record (souvent vide)
- Un superbloc :
  - Recopié au sein de la partition (en secours)
  - Il est composé :
    - Taille du système de fichiers
    - Nombre et liste de blocs libres ainsi qu'un index sur le prochain bloc libre
    - Taille de la liste des i-nodes, nombres d'i-nodes libres et index sur la prochaine i-node libre
    - Drapeau de modification du superbloc
- Une liste d'i-nodes
- Une liste de blocs de données

Les noeuds d'information : cas de Linux

Chaque i-node contient :

- Propriétaire du fichier (utilisateur et groupe)
- Type de fichier
- Permissions d'accès
- Dates d'accès (modification du fichier, accès au fichier, modification de l'i-node)
- Taille du fichier

Notion de Shell

Notion de Shell

- Une interface système ou shell en anglais (pour coquille ou coque) est une couche logicielle qui fournit l'interface utilisateur d'un système d'exploitation.
- Le shell peut être commandé de deux façon distinctes :
  - Une interface de commande (CLI pour Commande Line Interface en anglais) : le programme fonctionne alors à partir d'instructions en mode texte qui sont écrites dans un terminal (émulé ou non). Les manipulations s'y font rapidement.
  - Shell graphique fournissant une interface graphique pour l'utilisateur (GUI pour Graphical User Interface). Les manipulations sont intuitives.

Les différents Shell

- Sous Unix :
  - Bourne shell (sh), csh, tcsh, Korn shell (ksh) ou rc
  - Bourne-Again shell (bash) : est le plus répandu et s'inspire des versions ci-dessus
- Sous Windows : Shell regroupe l'interpréteur de commande et l'interface graphique (l'explorateur Windows)
  - Command.com pour les versions de Windows basées sur MS-DOS
  - cmd pour celles qui reposent sur Windows NT
  - Windows PowerShell pour Windows Vista (et utilisable pour Windows Vista)
- Sous Mac :
  - A l'origine tcsh, aujourd'hui bash

Forme générique d'une commande et manuel

La forme générique d'un commande texte est :

commande [-options] [liste d'opérandes]

Exemple : ls -l /tmp

- ls est la commande (afficher la liste des fichiers)
- -l est l'option (liste longue, avec les attributs des fichiers)
- /tmp est le nom de répertoire passé en argument (répertoire dont on liste les fichiers)

Remarques :

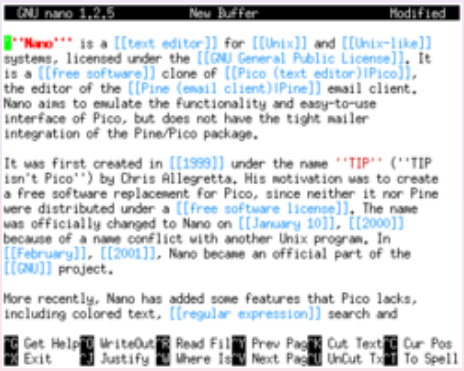
- si un nom de fichier commence par -, alors il peut y avoir confusion entre les options et l'argument. Dans ce cas, on place un double tiret après les options.
  Ex. ls -l – -fichier

Manuel des commandes

Une documentation existe pour chaque commande :

- man <commande> : permet d'accéder au manuel de la <commande>.
- Recherche dans man par mot-clé : man -k <mot-clé>
- info <commande> : permet d'accéder au manuel de la <commande>.
- <commande >–help
- man se trouve dans le répertoire sous forme compressée : /usr/share/man
- le caractère Q permet de sortir du man.
- recherche google avec les mots clés : <commande>linux

Définition



- Logiciel destiné à la création et l'édition de fichiers textes.
- Utile pour l'administration du système (fichiers de configuration) et le développement logiciel.
- Utile aussi en bureautique (Latex p.ex)

Fonctionnalités

Fonctionnalités :

- Ouvrir un fichier
- Ajouter du texte dans une ligne ou des lignes dans un fichier
- Ôter des caractères dans une ligne, ou des lignes dans un fichier
- Rechercher/remplacer des chaînes textes.
- Sauvergarder un fichier

Exemples

Exemples sous Unix/Linux :

- vi, Vim
- Nano, Pico
- Emacs

Exemples sous Unix/Linux (environnement bureau) :

- Kate sous l'environnement KDE
- gedit sous l'environnement GNOME
- Mousepad sous l'environnement Xcfe
- Leafpad sous l'environnement LXDE

Éditeur de texte : Vi et Vim

- vi est l'éditeur standard d'Unix. Favori des hackers jusqià l'arrivée d'emacs.
- vi est un éditeur modal c-à-d que la signification des touches change selon le mode dans lequel il se trouve.
- Vim "Vi IMproved" est un clone de vi et est généralement fourni avec les distributions linux.

Guide de survie avec vi(m) : les modes

- Ouvrir un fichier : vi <fichier >
- Trois modes :
  - Mode *commande* (dans lequel vi démarre)
  - Mode *insertion*
  - Mode *visuel*
- Changer de mode :
  - Par défaut le mode est le mode commande.
  - Pour passer en mode insertion, utiliser la commande **i**. La touche Echap permet de sortir du mode insertion.
  - Pour passer en mode visuel, utiliser la commande **v**. La touche Echap permet de sortir du mode insertion.

Guide de survie avec vi(m) : quitter et sauver

- Sortir et enregistrer (être en mode commande) :
  - **:w** enregistre le fichier courant
  - **:q** quitte vi (si tout est sauvé)
  - **:q !** quitte vi même si tout n'est pas sauvé
  - **:wq** enregistre le fichier courant et quitte

Guide de survie avec vi(m) : se déplacer et effacer

Pour se déplacer (être en mode commande) :

- **k** pour aller en haut
- **j** pour aller en haut
- **l** pour aller à droite
- **h** pour aller à gauche
- **gg** permet d'aller tout en haut.

Pour effacer une ligne ou un caractère (être en mode commande) :

- **x** permet d'effacer le caractère sous le curseur.
- **u** annule le dernier effacement
- **dd** permet d'effacer la ligne courante du curseur.

Remarque : les flèches en mode commande permettent parfois aussi de se déplacer.

Guide de survie avec vi(m) : Copier et coller

- Copier/coller :
  - Dans le mode visuel : déplacer le curseur pour sélectionner la zone à copier
  - Taper **"ay** pour désigner que l'on souhaite insérer dans le presse papier **a** ce qui vient d'être sélectionné. Ceci vous rebascule dans le mode commande.
  - Taper **"ap** pour copier ce qui est dans le presse papier à l'endroit du curseur. La zone sélectionnée est copiée.

Deuxième partie II

Description de l'arborescence Unix

Types de fichiers

En UNIX, tout est fichier. Il existe 6 types de fichiers :

Type de fichier	Fonction	Flag/type
Fichiers Standards	contient les données	-
Repertoires	Fichiers contenant les emplacements de fichiers et leurs noms	d
Fichiers "Device de /dev"	de type bloc (accès direct) de type caractère (accès séquentiel)	b c
Fichiers pipe	FIFO permettant la communication entre les processus	p
Fichiers socket. Ex : dans /tmp	points d'entréede communications entre processus basé sur les couches couche réseau	s
Liens	pointeurs des fichiers, peut être de type souple ou dur	l

Commande	Fonction
file <fichier>	permet de connaître le type d'un fichier

Notion d'arborescence

- Les répertoires (fichiers contenant des noms de fichiers) permettent de construire une arborescence logique.
- Un fichier UNIX peut être référencé par son chemin d'accès i.e. la description du chemin qu'il faut parcourir dans l'arborescence à partir d'un certain répertoire pour atteindre le fichier en question.
- / le répertoire qui désigne la racine racine absolue de l'arborescence.

Localisation d'un fichier/répertoire : Chemin relatif et absolu

- On distingue deux types de référencement d'un fichier/répertoire :
  - Chemin absolu : il comporte la liste complète des répertoires traversés depuis la racine et commence donc toujours par /
  - Chemin relatif : il comporte la liste des répertoire traversés depuis un répertoire spécifié.
- Le shell permet aussi de mémoriser un point de l'arbre que l'on appelle répertoire courant :
  - . désigne le répertoire courant de l'utilisateur
  - .. désigne le répertoire parent du répertoire courant de l'utilisateur

Description de l'arborescence

- Un système Unix est entièrement sauvergardé sur un système de fichiers qui est structuré sous la forme d'une arborescence grâce à la notion de répertoire.
- Les premières branches de cette arborescence peuvent être regroupées en "classes".

Description de l'arborescence (suite)

- Côté système :
  - Répertoire de boot
  - Répertoires indispensables pour l'armoçage de l'OS
  - Répertoires des points de montage
  - Répertoires liés au système
  - Répertoires des politiques de sécurité
- Côté utilisateur :
  - Répertoires des données
  - Répertoires des applications annexes
- Répertoires des données temporaires

Répertoire de boot

Répertoire	Contenu
/boot	stockage du (des) noyau(x) + fichier configuration du bootloader

Indispensable pour le démarrage de l'OS

Répertoire	Contenu
/	racine du système qui contient tous les répertoires
/etc	Fichiers de configuration et bd systèmes (mot de passe des utilisateurs..)
/dev	Fichiers spéciaux des (pseudo-)périphériques
/bin	BINaires des commandes de base (binutils) pour les utilisateurs et démarrer/réparer le système
/sbin	BINaires pour le Super-utilisateur (administrateur : root)
/lib	Bibliothèques systèmes

Répertoire des points de montage

Répertoire	Contenu
/mnt	Points de montage de système de fichiers
/media	Points de montage pour les médias amovibles (clé usb etc)

Répertoires liés au système

Répertoire	Contenu
/proc	Répertoire de système de fichiers virtuels : Informations systèmes (état du noyau et processus système)
/var	Données écrites par le système durant son exécution (log :/var/log, mail :/var/mail, impression :/var/spool, fichiers temporaires à conserver entre des reboot : /var/tmp, cache du gestionnaire de paquets :/var/cache/apt/archives etc)
/sys	Répertoire utile au système (point de montage interne, etc)

Répertoire des politiques de sécurité

Répertoire	Contenu
/selinux	Politiques de sécurité d'accès aux éléments de l'OS (démons ou fichiers)

Répertoires des données

Répertoire	Contenu
/home	Données des utilisateurs
/root	Données de l'administrateur
/srv	Données des services hébergés (site web, bd, etc). A tendance à remplacer certaines parties de /var

Répertoire des applications annexes

Répertoire	Contenu
/usr	(Unix system ressource) : exécutables du système non vitaux au démarrage et fonctionnement minimal. Par exemple, les programmes utilisateurs (/usr/bin), les bibliothèques (/usr/lib) , la documentation (/usr/share/doc,/usr/share/man et les fichiers temporaires utilisateurs (/usr/tmp). /user/local ressemble à /usr et sert pour les applications qui ne font pas partie du système.
/opt	Applications installées hors paquets (svt compilées). Alternative à /usr

Répertoires des données temporaires

Répertoire	Contenu
/tmp	Fichiers temporaires non nécessaires au reboot, souvent effacés durant celui-ci
/run	Fichiers temporaires non nécessaires au reboot (gestion des processus courants etc). Parfois système de fichiers virtuel.
/lost+found	Morceaux de fichiers (i-nodes) non recouvrables avec fsck

Troisième partie III

Variables d'environnement du Shell

Variables d'environnement

- Les variables d'environnement constituent un moyen d'influencer (et de l'adapter à l'utilisateur) le comportement des logiciels d'un système.
- Ces variables contiennent une valeur de type chaîne de caractères, dont la signification est propre à chaque variable.

Variables d'environnement
Définition
Variables locales et globales
Exemples de variables globales
Exemples de variables locales
Opérations sur variables d'environnement d'un Shell

Variables d'environnement

- Lorsqu'une application utilise le moyen d'authentification PAM, elle peut disposer des variables d'environnement de celui-ci (cfr. /etc/environment). On peut considérer ces variables d'environnement comme celles du système. C'est le cas par exemple de la variable PATH qui contient la liste des répertoires dans lesquels le système doit rechercher les programmes à exécuter.
- Chaque application peut définir et utiliser ses propres variables d'environnement.
- Les variables d'environnement d'une application gardent leur valeur durant toutes la session. Il faut les réinitialiser en début de session. On y reviendra dans le cas du Shell !

Variables d'environnement d'un Shell

- Le Shell dispose de deux types de variables d'environnement :
  - Les variables locales, propres à l'environnement de la session du shell en cours.
  - Les variables (globales) de l'environnment d'exécution qui seront transmises aux applications qui sont lancées depuis le Shell.

Quelques variables d'environnement globales d'un Shell

variable d'environnement	Contenu
HOME	contient le répertoire de l'utilisateur
USER	contient le login de l'utilisateur
PWD	contient le répertoire courant de l'utilisateur
SHELL	contient l'emplacement du Shell de connexion
PATH	contient la liste des répertoires où se trouvent les commandes que l'utilisateur peut exécuter

Quelques variables d'environnement locales d'un Shell

variable d'environnement	Contenu
HOSTNAME	contient le nom de la machine
HISTSIZE	contient la taille maximale des commandes exécutées contenues dans le fichier historique
PS1	Contient les paramètres d'affichage de l'invite de commandes du Shell
PS2	Contient le prompt

Opérations sur variables d'environnement d'un Shell

commande	Fonction
VAR=ch_caract	Affecte la chaîne de caractères **ch_caract** à la variable VAR
VAR=ch1ch2	Concatène les chaînes de caractères ch1 et ch2 et affecte le résulat à la variable VAR
echo $VAR	Affiche le contenu de la variable VAR sur la sortie standard
env ou printev	Affiche toutes les variables globales et leur contenu sur la sortie standard

Opérations sur variables d'environnement d'un Shell

commande	Fonction
set	Affiche toutes les variables locales et globales et leur contenu sur la sortie standard
export VAR	Fait passer une variable de l'environnement local du SHELL à son environnement d'exécution. La variable VAR sera disponible
unset VAR	Supprime une variable de l'environnement local du SHELL.

Quatrième partie IV

Manipulation de l'arborescence

Manipulation de l'arborescence : Se déplacer

Commande	Fonction	compte
pwd	Affiche le chemin absolu du répertoire courant	user
cd ..	Le père du répertoire courant devient le répertoire courant	user/root
cd CHILD	Le répertoire enfant CHILD devient le répertoire courant	user/root
cd TARGET	Le répertoir courant devient le répertoire spécifié par le chemin (absolu ou relatif) TARGET	user/root

Manipulation de l'arborescence : Information sur l'arborescence

Commande	Fonction	compte
ls	Lister les fichiers d'un répertoire	user
ls -a	Lister les fichiers d'un répertoire dont les fichiers cachés	user
ls -l	Lister les fichiers d'un répertoire en donnant beaucoup d'informations (attributs)	user
tree TARGET	Affiche l'arborescence de fichiers (et répertoires) à partir du répertoire TARGET spécifié (à installer)	user/root
find	Donne l'ensemble des chemins absolus de fichiers/répertoires à partir du répertoire courant.	user/root
find ROOT -name FILE	Donne l'ensemble des fichiers/répertoires à partir du répertoire ROOT contenant le fichier nommé FILE. Il y a beaucoup d'autres possibilités. Le format de la commade est : répertoire critère(s) action	user/root
du -h TARGET	Donne les tailles en ko,Mo etc de l'ensemble des répertoires (avec le chemin relatif) à partir de TARGET	user/root

Manipulation de l'arborescence : Création/Suppression et Mouvement

Commande	Fonction	compte
touch FILE	Modifie la date d'accès d'un fichier FILE, le crée si il n'existe pas	user/root
rm FILE	Efface le fichier FILE.	user/root
mv ORIGIN TARGET	Déplace le fichier. Il y a possibilité de renommer le fichier	user/root
cp ORIGIN TARGET	Copie le fichier	user/root
mkdir NAME	Crée un repertoire	user/root
rmdir NAME	Supprime un répertoire vide	user/root
rm -r NAME	Supprime un répertoire en le vidant de façon récursive	user/root

Liens symboliques

- Un lien symbolique est une entrée dans un répertoire qui permet de référenfer d'autres fichiers, répertoires etc. Lorsque l'on crée un raccourci sur le bureau, c'est en réalité un lien symbolique qui est créé.
- Si on supprime un lien symbolique, on ne fait que supprimer le chemin vers le fichier (ou répertoire). On ne supprime pas le bloc de données.
- Si on supprime le fichier original (et donc le bloc de donnée), le lien symbolique ne pointe plus vers quelque chose. On dit que le lien est cassé et souvent il change de couleur.
- Un lien symbolique est de type "l". Lorsque l'on tape "ls -l", il est indiqué vers quel fichier pointe le lien symbolique.

Liens symboliques (suite)

commande	Fonction
ln	commande qui fait des liens entre les fichiers
ln -s <fichier><lien_symb. >	crée un fichier de type "l" <lien symb. >contenant l'emplacement du <fichier >
rm <lien_symb.>	supprime le <lien symbolique>

Gestionnaire de fichiers léger Debian (LXDE) : PCManFM

Gestionnaire de fichiers

- But :
  - Se déplacer dans l'arborescence
  - Créer/Supprimer des fichiers et des répertoires
  - Rechercher de fichiers
  - Visualiser le contenu d'un répertoire
  - Visualiser les propriétés et permissions d'un fichier/répertoire
  - Trier les fichiers
  - ...
- Exemple pour Debian (LXDE) : PCManFM

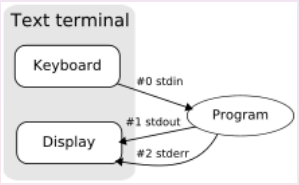
Cinquième partie V

Utiliser le Shell

Fichiers spéciaux d'entrée/sortie et d'erreurs

- Unix possède trois fichiers spéciaux d'entrée/sortie et d'erreurs :
  - /dev/stdin : fichier spécial contenant ce qui est entré au clavier.
  - /dev/stdout : fichier spécial recevant les données d'un programme en cours d'exécution. Ce fichier est généralement redirigé vers lécran correspondant au tty en avant-plan.
  - /dev/stderr : fichier spécial recevant les messages d'erreur d'un programme en cours d'exécution. Ce fichier est généralement redirigé vers lécran correspondant au tty en avant-plan.

Accéder aux fichiers spéciaux d'entrée/sortie et d'erreurs via une commande Shell



Dans le Shell, on peut faire référence à ces fichiers par les chiffres suivants :

- 0 : (entrée standard) correspond au fichier /dev/stdin
- 1 : (sortie standard) correspond au fichier /dev/stdout
- 2 : (sortie d'erreur standard) correspond au fichier /dev/stderr

Il est possible de rediriger cette entrée et ces sorties !

Éditeur de texte pour le shell

commande	Fonction
cat <fichiers>	Affiche à l'entrée standard le contenu des fichiers reçus en argument. Si aucun fichier n'est spécifié, cat considère l'entrée standard.
more <fichier>	Action similaire à cat en permettant de visualiser l'entrée standard page par page.
less <fichier>	Action similaire à more.

Redirection de la sortie standard vers un fichier (>et >>)

- **commande** >fichier : redirige la sortie standard (1) de la **commande** dans le fichier. Le fichier contient la sortie standard de la <commande >et rien de plus (même si il contenait des données)
- **commande** >>fichier : redirige la sortie standard (1) de la **commande** à la fin du fichier.

Exemples :

- ls -l >Liste
- moyenne.out >>resultats

Redirection de la sortie d'erreurs vers un fichier (2>et 2>>)

- **commande** 2>fichier : redirige la sortie d'erreurs (2) de la **commande** dans le fichier. Le fichier contient la sortie standard de la <commande >et rien de plus (même si il contenait des données)
- **commande** 2>>fichier : redirige la sortie d'erreurs (2) de la **commande** à la fin du fichier.

Redirection de la sortie d'erreurs vers un fichier (2>et 2>>) (suite)

Exemples :

- find /usr/share nano 2>fichier-erreur.txt : redirige les messages d'erreurs dans le fichier-erreur.txt
- find /usr/share nano 2>/dev/null : redirige les messages d'erreur dans le fichier /dev/null (”poubelle”)
- find /usr/share nano 2>>erreurs.txt : redirige les messages d'erreur à la fin du fichier erreur.txt

Redirection de la sortie d'erreurs et de la sortie standard vers des fichiers

- **commande** 1>file-std 2>file-erreurs : redirige la sortie standard (1) de la commande dans le fichier file-stad et la sortie d'erreurs de la commande dans le fichier file-erreurs. Les fichiers ne contiennent que ce qui a été envoyé et rien de plus (même si il contenait des données)

Exemple : ls 1>file-std 2>file-erreurs

Redirection de l'erreur standard vers la sortie standard (2 >&1)

- **commande** fichier 2 >&1 : regroupe la sortie d'erreur et la sortie standard en un flux envoyé sur la sortie standard

Exemples :

- cat fichier_existant fichier_inexistant : affiche le fichier existant et un message d'erreur
- cat fichier_existant fichier_inexistant >concatenation.txt : affiche un message d'erreur et envoie le fichier existant dans le fichier concatenation.txt
- cat fichier_existant fichier_inexistant >concatenation.txt 2>&1 : n'affiche plus rien et envoie le contenu du fichier existant et le message d'erreur dans le fichier concatenation.txt

Redirections de l'entrée depuis un fichier (<)

- **commande** <fichier : la **commande** prend le fichier comme argument

Exemples :

- redirection du fichier file vers l'entrée standard : cat <file

Groupement de commandes

Il est possible de grouper plusieurs commandes sur une seule ligne. Chaque ligne de commande est appelé un job (qui est un groupe de processus ; voir plus loin).

commande1 ; commande2 ; commande3

Les commandes vont être exécutées les unes après les autres en affichant leurs messages sur la sortie standard et d'erreurs.
Exemple :

- ls ; pwd ; date

Groupement de commandes (parenthésage)

Il est possible de regrouper certains commandes en utilisant des parenthèses. Dans ce cas, ce qui se trouve à l'intérieur de la paranthése est exécuté dans un sous-Shell :

(commande1 ; commande2) ; commande3

Exemple :

- (cd .. ; pwd) ; pwd : ouvre un sous-shell identique au shell courant, remonte d'un répertoire, affiche celui-ci. Affiche le répertoire du Shell principal.
- Notons la différence avec cd .. ; (pwd ; pwd)

Groupement de commandes (lancement conditionnel)

Il est possible de lancer des commandes uniquement si un évènement se produit.

- commande 1 && commande2 : commande2 sera lancé uniquement si commande1 n'a pas envoyé de message d'erreur
- commande 1 || commande2 : commande2 sera lancé uniquement si commande1 a envoyé un message d'erreur

Tubes ou pipes

Pour exécuter des commandes qui utilisent chacunes le résultat de la précédente, une solution serait d'utiliser un fichier temporaire :

commande1 >fichier
commande2 <fichier "supprimer" le fichier

Une solution plus efficace et plus simple à mettre en oeuvre, surtout lorsqu'il y a plus de deux commandes, consiste à injecter la sortie standard d'une commande dans l'entrée standard de la suivante. C'est le mécanisme du tube (ou pipe) :

commande1 | commande2 | commande3

Exemple : ls | more

Commande Tee

- **tee** fichier : duplique l'entrée standard de la fonction **tee** sur sa sortie sortie standard et sur le fichier fourni en argument
- **Exemples :**
  - Cette fonction est surtout intéressante pour les pipes afin d'afficher les résultats intermédiaires. Exemple :

    commande1 | tee resultat_int| commande2

Filtres

Notion de filtre

Un filtre est une fonction qui prend généralement entrée (standard ou via un fichier) une liste d'éléments et qui fournit une sélection de celles-ci en fonction d'un certain critère.

Filtres (suite)

grep [opt.] MOTIF <FICHIER >

Grep recherche dans les FICHIERs indiqués (ou depuis l'entrée standard si aucun fichier n'est fourni, ou si le nom "tiret", i.e - ,est mentionné) les lignes comprenant un certain MOTIF et les écrit dans la sortie standard.

Entrée standard, sortie standard et sortie d'erreurs
Éditeur de texte pour le shell
Redirections
Ensemble de commandes
Initialisation du Bash
Groupement de commandes
Lancement conditionnel
Tubes ou pipes
Commande Tee
Filtres

Filtres (suite)

head <FICHIER >

head recherche dans les FICHIERs indiqués (ou depuis l'entrée standard si aucun fichier n'est fourni, ou si le nom "tiret", i.e - ,est mentionné) les 10 premières lignes les écrit dans la sortie standard.

Filtres (suite)

tail <FICHIER >

tail recherche dans les FICHIERs indiqués (ou depuis l'entrée standard si aucun fichier n'est fourni, ou si le nom "tiret", i.e - ,est mentionné) les 10 dernières lignes les écrit dans la sortie standard.

Remarque : Les filtres prennent toujours des fichiers qui se termine par un symbole de fin de fichier (EOF) qui est le caractère Ctrl et dans être en début de ligne.

Méta-caractères

Méta-caractère	Fonction
?	représente n'importe quel caractère
*	représente n'importe quelle chaîne de caractères
[ensemble]	représente n'importe quel caractère appartenant à l'ensemble. Options : • si l'ensemble est 123, cela signifie les éléments 1, 2 et 3 • si l'ensemble est !12, cela signifie les éléments qui ne sont pas 1 ou 2 • si l'ensemble est 1-3 : cela signifie les éléments de 1 à 3 : 1,2 et 3 • si l'ensemble est !1-3 : cela signifie les éléments de qui ne sont ni 1 ni 2 ni 3 • si l'ensemble est b-d : cela signifie les éléments de b à d : b, c et d

Initialisation du Bash

- Il est possible de demander à Bash d'exécuter des commandes ou initialiser des variables d'environnement lors de son lancement.
-
  - /etc/bash.bashrc configure tous les bash de la machine.
  - ∼.bashrc est lu pour l'utilisateur du home dans lequel il se trouve.

Les alias

- Les alias permettent de changer le nom d'une commande.
- Lorsqu'on utilise souvent une commande avec des options, cela vaut la peine d'utiliser un alias.
- Afin de conserver ces alias, il est conseillé de les mettre dans les fichiers d'initialisation des bash.

alias nom_de_votre_alias="commande de votre alias"

Source des images

- Wikipedia