SYSTEME D'EXLOITATION
Linux – Shell Bash
Programmation de shell

Un « Shell » est un interpréteur de commandes du système UNIX. C'est aussi un langage de commande, qui réalise l'interface entre l'utilisateur et le noyau d'UNIX. Les commandes exécutées par un Shell peuvent provenir soit du terminal, soit d'un fichier appelé fichier de commandes.
Le Shell est un véritable langage de programmation, c'est à dire, il possède :

- la notion de variable,
- des structures de contrôle (if, for while, ..)

Ce document vous présente les commandes de base du bash. Ce document n' a pas pour but de vous donner l'ensemble des commandes unix.... mais simplement de vous orienter pour une première utilisation.
Pour vous aider, une commande essentielle: man (pour manuel).
Exemple : `man ps` vous affichera les explications de fonctionnement de la commande ps
Dans les instructions ci-dessous, les mots en *italique* sont à remplacer par des éléments qui conviennent

1. La gestion des fichiers et répertoires

Copie de fichier	
cp *fichier1 fichier2*	Copie fichier1 vers fichier2
cp *fichier /dossier*	Copie fichier1 vers le répertoire /dossier
cp *dossier1 dossier2*	Copie le répertoire dossier1 (et les fichiers qu'il contient) vers le répertoire dossier2
cp -r *dossier1 dossier2*	Copie le répertoire dossier1 (et les fichiers et dossiers qu'il contient) vers le dossier2
Renommage et déplacement de fichier	
mv *fichier1 fichier2*	Renomme fichier1 en fichier2
mv *dossier1 dossier2*	Renomme dossier1 en dossier2
mv *fichier dossier*	Déplace fichier dans dossier
mv *fichier1 dossier/fichier2*	Déplace fichier1 dans le répertoire dossier et le renomme en fichier2
Suppression de fichier	
rm *fichier*	Supprime fichier du répertoire courant
Création de répertoire	
mkdir *dossier*	Crée le répertoire dossier
mkdir -p *dossier1/dossier2*	Crée des répertoires imbriqués

Suppression de répertoire	
rm -d *dossier*	Supprime le répertoire dossier
rm -rf *dossier*	Supprime le répertoire dossier et son contenu
Divers	
ln -s *fichier raccourci*	Crée un raccourci vers fichier
find *dossier* -name *fichier*	Cherche fichier dans dossier et ses sous répertoires
diff *fichier1 fichier2*	Compare 2 fichiers ou répertoires
sort *fichier1 fichier2*	Trie les lignes de tous les fichiers et les affiche à l'écran
Visualisation du contenu d'un fichier	
cat *fichier*	Affiche le contenu de fichier à l'écran en ASCII
more *fichier*	Affiche fichier à l'écran : « entrer » → descend d'une ligne, « espace » → descend d'une page, q = quitte
less *fichier*	Comme more, mais on peut utiliser la touche [Page Suivante]
head -n *fichier*	Affiche les n premières lignes de fichier
tail -n *fichier*	Affiche les n dernières lignes de fichier
vi *fichier*	Édite fichier avec l'éditeur vi
nano *fichier*	Édite fichier avec l'éditeur nano
gedit *fichier*	Édite fichier avec l'éditeur gedit
Gestion du contenu d'un fichier	
grep *chaîne fichier*	Affiche les lignes de fichier contenant chaîne
grep -r *chaîne dossier*	Recherche chaîne à travers tous les fichiers d'un répertoire
command > *fichier*	Met dans fichier la sortie / le résultat habituellement affiché à l'écran de command
command >> *fichier*	Ajoute dans fichier la sortie / le résultat habituellement affiché à l'écran de command
Contenu d'un répertoire	
ls -l *dossier*	Liste le contenu du répertoire dossier en mode détaillé
ls	Liste le contenu du répertoire courant
ls -a	Liste tous les fichiers (y compris les fichiers cachés)
ls -d	Liste les répertoires contenu dans le répertoire courant
ls -l | more	Liste le contenu du répertoire par page grâce à more
Déplacement dans l'arborescence des répertoires	
pwd	montre le nom du répertoire courant
cd *dossier*	Se déplace dans le répertoire dossier
cd	Se déplace dans le dossier /home/utilisateur (dossier « racine » du compte utilisateur

cd ..	Se déplace dans le dossier parent (situé au dessus) du répertoire courant
Permission et droit des fichiers	
chown *utilisateur fichier*	Rend utilisateur le propriétaire de fichier
chown -R *utilisateur dossier*	Rend utilisateur le propriétaire de dossier et de ses sous répertoires (-R)
chgrp *groupe fichier*	Change fichier afin qu'il appartienne à groupe
chmod u+x *fichier*	Donne (+) le droit d'exécuter (x) à l'utilisateur (u)
chmod g-w *fichier*	Retire (-) le droit d'écrire (w) au groupe (g)
chmod -R a+rx dossier	Donne (+) à tout le monde (a) les droits de lecture (r) et d'ouverture (x) de dossier de tous ses sous répertoires (-R)

2. Le contrôle des processus

ps -ef	Affiche tous les processus exécutés (pid et ppid)
ps aux	Affiche tous les processus, avec un maximum de détails
ps aux | grep *nomprocess*	Affiche les processus en rapport nomprocess
kill *pid*	Envoie un signal d'arrêt au processus pid
kill -9 *pid*	demande au système de tuer le processus pid

3. Manipulation des archives

tar xvf *archive.tar*	Extrait les fichiers archive.tar, en affichant les noms des fichiers
tar xvfz *archive.tar.gz*	Extrait les fichiers de l'archive en utilisant gzip puis tar
tar jxvf *archive.tar.bz2*	Extrait les fichiers de l'archive en utilisant bzip2 puis tar
tar cvf *archive.tar fichier1 fichier2*	Crée un fichier archive.tar contenant fichier1, fichier2.
tar cvfz *archive.tar.gz dossier*	Crée un fichier gzip contenant le répertoire dossier
gzip *fichier.txt*	Crée le fichier fichier.txt.gz
gunzip *fichier.txt.gz*	Extrait le fichier fichier.txt
bzip2 *fichier.txt*	Crée le fichier fichier.txt.bz2
bunzip2 *fichier.txt.bz2*	Décompresse le fichier fichier.txt.bz2.

4. Commandes diverses

date	Affiche la date courante
echo	Envoi en écho des paramètres spécifiés à la suite ($PATH, "bille")

id	Retourne le nom de login, l'UID, le groupe et le GID
logname	Retourne le nom de login de l'utilisateur
man *nomcommande*	Retourne le mode d'emploi de la commande s'il existe
sleep *n*	Ne fait rien pendant n secondes
touch *fichier*	Crée le fichier « fichier » s'il n'existe pas déjà. Si fichier existe déjà, modifie les caractéristiques d'un fichier (option -a : date du dernier accès)
uname	Donne le nom du système d'exploitation
who	Retourne le nom des utilisateurs qui ont ouvert une session
who am i	Retourne le nom de l'utilisateur de la session courante
groups *utilisateur*	Affiche à quel groupe appartient utilisateur

5. Re-direction des entrées / sorties

Lors de l'ouverture d'une session, l'utilisateur possède, par l'intermédiaire de son terminal :

- une entrée standard définie par l'identificateur stdin et correspondant à l'entier 0, par défaut c'est le clavier,
- une sortie standard définie par l'identificateur stdout et correspondant à l'entier 1, par défaut c'est l'écran,
- une sortie d'erreur définie par l'identificateur stderr et correspondant à l'entier 2, par défaut c'est l'écran.

Chaque commande ou programme utilisateur peut utiliser cette entrée ou ces sorties. Par exemple, les commandes suivantes :

- `who` utilise stdout pour afficher la liste des utilisateurs logés,
- `cc` (le compilateur c) utilise stderr pour afficher les éventuels messages d'erreur

Linux permet la re-direction des entrées et sorties des commandes. Pour cela on utilise les caractères > et <.

- Le caractère > redirige la sortie de la commande. Il est possible d'ajouter la sortie d'une commande à un fichier déjà existant avec >>.
  Exemple : `ls > Liste.txt` : la liste des fichiers est re-dirigée vers le fichier Liste.txt

- Le caractère < redirige l'entrée standard d'une commande.
  Exemple : `monprogramme < données.txt`
  si l'exécutable « monprogramme » nécessite des données à saisir, ces données peuvent être placées dans le fichier « données.txt ». L'exécution de « monprogramme » ne nécessitera plus la saisie des données.

- Pour rediriger vers la sortie d'erreur, il suffit d'utiliser le n° qui concerne la sortie d'erreur (2)
  Exemple: `monprogramme 2 > erreurs.txt`

6. Enchaînement des commandes

Il est possible d'enchaîner l'exécution des commandes à l'aide de l'opérateur ; (point virgule). Chaque commande est alors exécutée séquentiellement.

Exemple : `date;who;ls`

7. Connexion de processus, les tubes ou pipes

Chaque processus peut disposer de l'entrée stdin des sorties stdout et stderr. On peut donc rediriger l'entrée ou la sortie standard d'une commande. Il est aussi possible de relier la sortie standard d'un processus à l'entrée standard d'un autre processus. Si par exemple on veut connaître à un instant donné le nombre d'utilisateurs loggés sur le système, on exécutera les commandes who et wc de la façon suivante:

`who > temp.txt` Le fichier temp.txt contient le résultat de la commande who qui liste les utilisateurs connectés.

`wc -l < temp.txt` La commande wc avec l'option l compte et affiche le nombre de lignes du fichier « temp.txt ». On obtient donc le nombre d'utilisateur connectés, le fichier temp.txt permet la « connexion » entre les deux commandes.

Il existe un opérateur **tube** ou **pipe** qui réalise la même fonction et évite la création du fichier intermédiaire. C'est l'opérateur « | » qui relie la sortie standard d'un processus à l'entrée standard d'un autre processus. Exemple : `who | wc -l`

8. Les caractères spéciaux

Certains caractères ont une signification particulière pour le Shell et sont appelés caractères spéciaux ou métacaractères , ce sont : *, ?, [...],$, #, &, ; ,> , >>, <, <<, |, ‘,",`, \, (), {}.

Les métacaractères *, ?, [...] permettent de construire des chaînes de caractères génériques, ainsi :

- * désigne une chaîne de caractères quelconque ;
- ? désigne un caractère quelconque ;
- [...] désigne les caractères entre crochets, définis par énumération ou par un intervalle.

D’autres métacarcatères permettent de modifier l'interprétation d'une commande :

- ; (le point virgule) sépare deux commandes sur une même ligne ;
- ' (l’apostrophe) délimite une chaîne de caractères contenant des espaces (à l'intérieur, tous les métacaractères perdent leur signification) ;
- " (le guillemet) délimite une chaîne de caractères contenant des espaces (à l'intérieur, tous les métacaractères perdent leur signification, à l'exception des métacaractères `, \et $) ;
- ` (l’accent grave) "capture" la sortie standard pour former un nouvel argument ou une nouvelle commande (cette particularité est présentée plus loin) ;
- \ (l’anti-slash) annihile la signification du métacaractère qui suit ;
- { et } (les accolades) permettent de regrouper un ensemble de commandes et de les exécuter dans le "Shell courant" ;
- (et) (les parenthèses) permettent de regrouper un ensemble de commandes et de les exécuter dans un "Shell fils".