Unix/Linux utilisateur, l’essentiel

Sébastien Jeudy

(Version V1)

Unix/Linux utilisateur, l’essentiel

Objectif et sommaire :

Acquérir les connaissances essentielles d’utilisation d’un système d’exploitation Unix / Linux en ligne de commande :

I. Prise en main du système
II. Gérer les fichiers, utiliser les éditeurs
III. Comprendre et utiliser le Shell
IV. Le fonctionnement multitâche et les processus
V. Unix/Linux en réseau local

I. Prise en main du système

1) Historique des systèmes Unix/Linux
2) Les différentes versions d'Unix (AIX, HP-UX, Solaris,…)
3) Les distributions Linux
4) Architecture du système
5) Utilisation du système et interface graphique
6) Structure d'une ligne de commande
7) Arborescence standard du système de fichiers Unix
8) Organisation des répertoires et des fichiers
9) Utilisateurs et groupes, protections d'accès aux fichiers

II. Gérer les fichiers, utiliser les éditeurs

1) Documentation en ligne de commande
2) Les commandes de base pour les fichiers et les répertoires
3) Les liens physiques et symboliques
4) Commandes complémentaires
5) Les différents éditeurs
6) L’éditeur vi

III. Comprendre et utiliser le Shell

1) Notion de Shell
2) Les variables
3) L’expansion/substitution de commandes
4) Les alias
5) Les filtres
6) Les redirections
7) Notion de scripts Shell

IV. Le fonctionnement multitâche et les processus

1) Exécution en arrière plan
2) Outils pour le background
3) Signaux et suppression d’un processus

V. Unix/Linux en réseau local

1) Notions sur TCP/IP
2) Commandes distantes sécurisées (ssh)

I. Prise en main du système

I. Prise en main du système

1) Historique des systèmes Unix/Linux
2) Les différentes versions d'Unix (AIX, HP-UX, Solaris,…)
3) Les distributions Linux
4) Architecture du système
5) Utilisation du système et interface graphique
6) Structure d'une ligne de commande
7) Arborescence standard du système de fichiers Unix
8) Organisation des répertoires et des fichiers
9) Utilisateurs et groupes, protections d'accès aux fichiers

1) Historique des systèmes Unix/Linux

1965 : MULTICS, OS simple par Ken Thompson (Bell AT&T)

1969 : UNICS, version réduite de MULTICS (Thompson&Ritchie)

1970 : UNIX officiel (horloges systèmes > 01/01/1970)

1973 : UNIX TSS, réécrit en Langage C (Kernighan&Ritchie)

1977 : BSD, UNIX réécrit pour plates-formes VAX (Berkeley)

80's : UNIX like (AIX / IBM, Sun Solaris / Sun Microsystems, HP-UX / Hewlett-Packard,…)

1983 : GNU (GNU's Not UNIX)

1991 : Linux pour x86 par Linus Torvalds (étudiant finlandais)

1999 : Mac OS X puis iOS par Apple (origine NeXT)

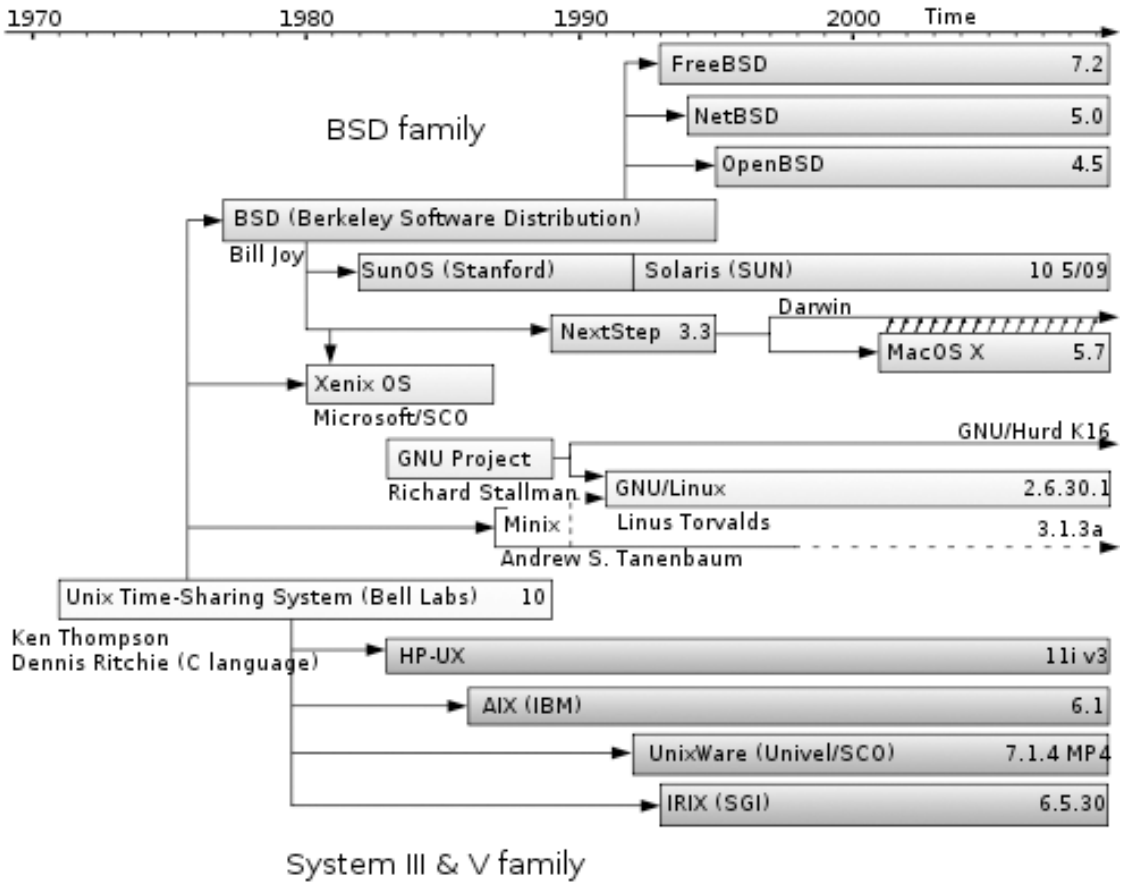
2007 : Android pour smartphones et tablettes (Google)

Notes :

http://fr.wikipedia.org/wiki/Unix

De l’open-source (payant) au librement redistribuable (gratuit) :
L’ouverture des codes sources UNIX par AT&T a largement contribué, très tôt (1977), à sa popularité. Encore davantage avec sa licence devenue, plus tard, librement redistribuable.

2) Les différentes versions d'Unix (AIX, HP-UX, Solaris,...)



3) Les distributions Linux

Linux, ou GNU/Linux, est un **système d'exploitation respectant les normes POSIX** (standards UNIX).

Linux est fondé sur le noyau Linux (v3.0 au 22 juillet 2011), **logiciel libre créé en 1991 par Linus Torvalds** pour ordinateur compatible PC.

La **licence publique générale GNU**, ou GNU General Public License en anglais (GNU GPL), est une licence qui fixe les conditions légales de distribution des logiciels libres du projet GNU.

Notes :

http://www.linuxfoundation.org

https://www.kernel.org

http://fr.wikipedia.org/wiki/Linux

http://fr.wikipedia.org/wiki/GNU

Une distribution Linux est un ensemble cohérent de logiciels (la plupart libres) assemblés autour du système d'exploitation Linux.

Les distributions majeures (plutôt entreprises) :

Debian : distribution non commerciale.

Slackware : l'une des plus anciennes distributions.

SuSE : la plus ancienne distribution commerciale.

Red Hat : basée sur les paquets RPM (+ autres distributions).

Des distributions communautaires et grand-public :

OpenSUSE : version grand-public de SuSE.

Fedora, CentOS : versions communautaires de Red Hat.

Gentoo : paquetages à la manière des ports BSD.

Ubuntu : basée sur Debian (grand-public & entreprise).

Notes :

http://fr.wikipedia.org/wiki/Distribution_Linux

4) Architecture du système

Basé sur un noyau (kernel) : cœur du système d'exploitation

Système d'exploitation multitâches : programmes ou processus gérant :

Matériel (hardware) <=> UNIX (OS) <=> Applications (software)
"en parallèle" (processus, mémoire, fichiers, périphériques, réseau,…)

Système d'exploitation multi-utilisateurs : plusieurs utilisateurs sur le même système central via des terminaux (ou des interfaces graphiques)

Système de fichiers : sur Unix / Linux tout est fichier (3 types : ordinaires txt / bin, répertoires, spécifiques aux périphériques)

Gestion des tâches : processus / process = programme en cours ou en attente d'exécution en mémoire (indépendant, multitâche)

Identifié par un **numéro unique PID** (16 bits) + Informations en mémoire (table des processus) : commande « ps »

Le Shell : interpréteur de commandes & langage scripts

Interface entre l'utilisateur et le système d'exploitation (terminal)
Différentes versions avec spécificités (sh, bash, ksh, csh,…)

Outils fournis : programmes & utilitaires pour utiliser l'OS

(commandes : ls, mkdir, grep, ps, sh, cal, man, mail, vi, tar,…)

5) Utilisation du système et interface graphique

Ouverture d'une session :

Accéder au système avec un (ou des) compte(s) utilisateur(s)
Plusieurs sessions possibles (indépendantes et multitâches)

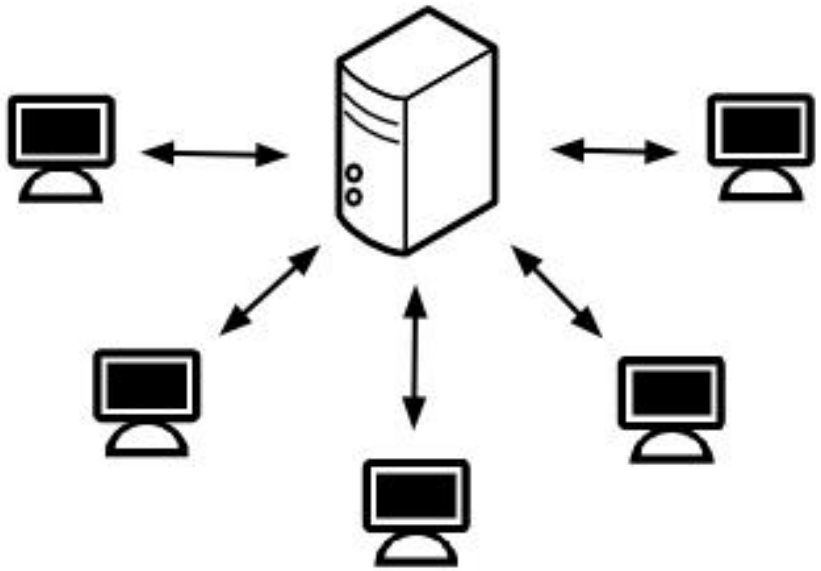
En local : UNIX + Applications installés et exécutés sur l'ordinateur utilisé (via interface graphique ou console texte)

Sur une machine distante (host) : UNIX + Applications installés et exécutés sur l'ordinateur distant (= le serveur)

Accès par un terminal ou par une émulation de terminal (= le client, ex : telnet ou ssh) via un réseau série / LAN / Internet (TCP/IP)

Notes :

Système « client-serveur » :



Exemple de connexion par telnet (similaire avec ssh) :

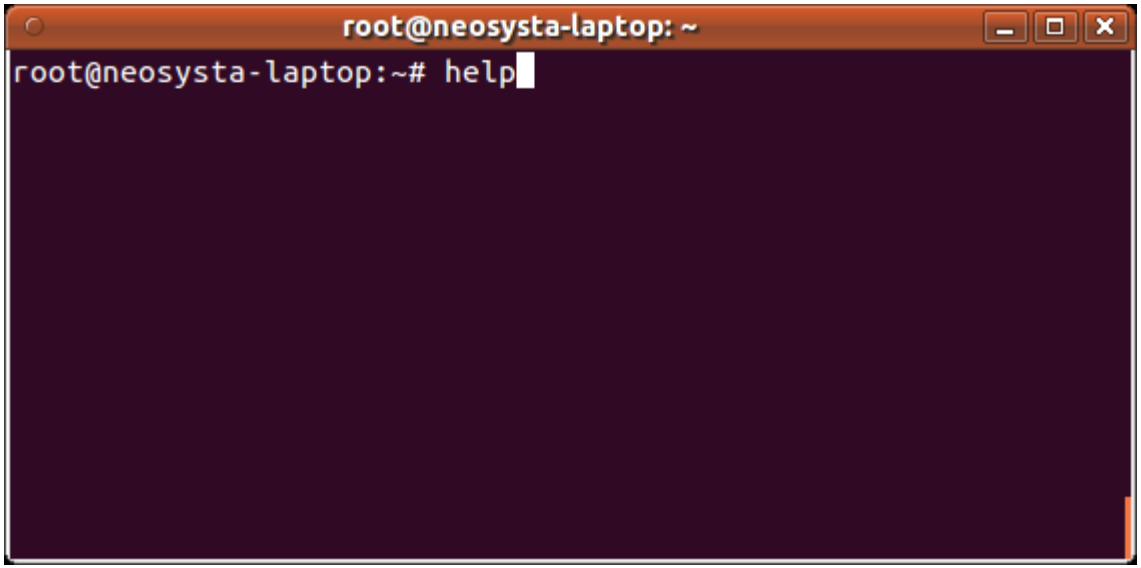
telnet 192.168.10.200
login : Ldupont
password : mypassword
$

Login : identifiant utilisateur connu du système pour se connecter (unique)

Password : mot de passe de connexion pour ce login

Notes :

Terminal Shell (lancé depuis l'interface graphique) :



Invite ou prompt (Shell) :

Login > Connexion > Shell > Prompt d'attente d'une commande

Symbole : $ ou # ou > ou % ou autre

Fermeture d'une session (Shell) : CTRL+D « exit » « logout »

Arrêt complet du système :

Uniquement par l'utilisateur « root » (super-utilisateur / administrateur) en ligne de commande : « halt » « shutdown -h now » « init 0 »

Arrêt de tous les services et programmes + déconnexion de tous les utilisateurs (!)

Redémarrer le système : « reboot » « init 6 »

L’interface graphique (ou Interface Homme-Machine) :

L’interface graphique est un ensemble de programmes Unix/Linux parmi les autres, permettant un **accès plus visuel et ergonomique au système** : bureau avec fenêtres, icônes, menus, usage de la souris,…

Généralement lancée au démarrage, son utilisation reste similaire d’un système Unix/Linux à l’autre, même si sa forme est différente (options et design). Plusieurs interfaces graphiques existent et peuvent cohabiter.

De nombreux outils avec interface graphique appellent directement les commandes du système (à commencer par le Terminal Shell).

Exemples d’interfaces graphiques Unix/OS X/Linux :
X-Window, Aqua, GNOME, KDE, Xfce,…

Notes :

En anglais, **GUI** est l’abréviation de Graphical User Interface, soit « interface graphique utilisateur ». Elle s’oppose à **CLI** pour Command Line Interface, soit « interface en ligne de commande ».

Remarque : les commandes « startx » et « xinit » permettent de démarrer une interface graphique depuis un terminal Shell en mode texte.

6) Structure d’une ligne de commande

Invite ou prompt (Shell) :

Terminal > Shell > Prompt d’attente d’une commande

Symbole : $ ou # ou > ou % ou autre

Syntaxe générale d’une commande : cde opt(s) arg(s)

Commande : nom de fonction, script exécutable, programme,…
Option(s) : caractère(s) précédé(s) par ‘-’ (cas de traitement)
Argument(s) : donnée(s) traitée(s) par la commande

Message d’erreur si syntaxe incorrecte (attention : casse respectée)

Notes :

Option(s) et Argument(s) ne sont pas toujours présents ou obligatoires. Ils dépendent de la commande et de son contexte d’utilisation.

Cycle d’exécution d’une commande :

1) L’utilisateur saisit la commande (invite / prompt) + ENTREE
2) Le Shell vérifie la syntaxe et recherche la commande :
   commande interne (Shell), sinon commande externe ($PATH)
3) Si OK, le programme de la commande est exécuté (durée variable)
4) La fin du programme retourne au Shell (invite / prompt)

Interruption de l’exécution d’une commande :

Interrompre un programme : CTRL+C (CTRL+D)

Les commandes principales Unix/Linux sont intégrées au Shell (commandes internes) et dans les répertoires système (commandes externes) : « help » « ls /bin/ » « ls /sbin/ »

Notes :

« $PATH » est une variable système qui contient un ensemble de chemins de répertoires (ordonnés et séparés par ‘:’) dans lesquels le Shell recherche les programmes/commandes externes à exécuter (fichiers exécutables). Dès que le programme est trouvé dans un répertoire, celui-ci est exécuté.

Notes :

Mise en pratique : tester la commande « ls »

ls

ls -l

ls -a

ls -l -a

ls -a -l

ls -la

ls -al

ls /bin/

ls -la /sbin/

7) Arborescence standard du système de fichiers Unix

Le système de fichiers : structure d'accueil des données sur un support physique (D7, HD, CD, DVD, USB,…) définie lors du formatage.

Constitution du système de fichiers Unix/Linux : composé…

Des fichiers ordinaires : suites d'octets txt / bin
Des répertoires : fichiers listes de noms avec identifiants inodes
Des fichiers spéciaux : références à des périphériques
D'une table d'inodes : références à tous les fichiers + informations
D'un super bloc : bloc de données en début de partition (informations sur le système de fichiers, taille de la table d'inodes, taille des blocs,…)

Arborescence des fichiers sous Unix / Linux :

Organisation selon une structure arborescente (ensemble de répertoires contenant d'autres répertoires et les fichiers, sous forme d'arbre)

Répertoire de départ : racine = 'root' (en anglais) = '/'

1 seule racine pour tout le système

(≠ MS-DOS/Windows : 1 racine par lecteur / partition de disque)

Le FHS (Filesystem Hierarchy Standard) :

Norme d'uniformisation des emplacements et noms des fichiers + répertoires dans les arborescences UNIX

Notes :

Lister les répertoires et fichiers à la racine : ls -la /

Lister toute l'arborescence système depuis la racine : ls -R /

/ : Racine du système

/bin : Exécutables des commandes essentielles

/boot : Fichiers statiques du chargeur d'amorçage

/dev : Fichiers spéciaux des périphériques

/etc : Fichiers de configuration

/home : Répertoires personnels des utilisateurs

/lib : Bibliothèques partagées essentielles et modules du noyau

/media : Contient les points de montages pour les médias amovibles

/mnt : Point de montage pour monter temporairement un système de fichiers

/proc : Répertoire virtuel pour les informations système (noyau)

/root : Répertoire personnel du super-utilisateur (administrateur)

/sbin : Exécutables système essentiels

/srv : Données pour les services du système

/tmp : Fichiers temporaires

/usr : Hiérarchie secondaire

/var : Données variables et diverses

/opt, /usr/local : Paquets pour applications supplémentaires

Notes :

A l’ouverture d’un Shell, l’utilisateur se trouve par défaut dans son répertoire de travail (ou dossier personnel) :

/home/utilisateur/

8) Organisation des répertoires et des fichiers

Noms des fichiers : lettres minuscules et majuscules (attention : casse respectée), chiffres et caractères spéciaux.

‘/’ interdit ; ‘-’ interdit au début ; ‘.’ cache le fichier si au début

Chemin : noms des répertoires séparés par ‘/’ pour accéder à un répertoire

Chemin absolu : chemin depuis la racine ‘/’
Chemin relatif : chemin depuis le répertoire courant

Lister le contenu d’un répertoire : ls chemin (absolu ou relatif)

Caractéristiques des fichiers & répertoires :

Type du fichier : défini par un attribut (caractère de gauche par « ls -la »)

Attributs : ‘-’ (fichier standard) ; ‘d’ (répertoire) ; ‘l’ (lien symbolique)

Autorisations d’accès au fichier : gérées par une combinaison de 3 x 3 caractères (voir « ls -la »)

Autorisations par 3 catégories : propriétaire (u) / groupe (g) / autre (o)
Droits par catégorie : - (sans) / r (read) / w (write) / x (exéc. ou répertoire)

Nombre de sous-répertoires pour les répertoires (ou de liens phys.)

Nom du propriétaire et du groupe pour le fichier

Taille du fichier (en octets)

Date et heure de dernière modification sur le fichier

Nom du fichier ou du répertoire (‘.’ = caché)

Notes :

Lister le contenu du répertoire courant : ls -la

```
drwxr-xr-x  3 neosysta neosysta  4096 mars   2  2011 .
drwxr-xr-x 68 neosysta neosysta  4096 avril  8 19:55 ..
-rw-r--r--  1 neosysta neosysta    13 mars   2  2011 fichtest
drwx------  3 neosysta neosysta 12288 mars   6  2011 rdiff-backup-data
```

« **.** » correspond au répertoire courant

« **..** » correspond au répertoire parent

9) Utilisateurs et groupes, protections d'accès aux fichiers

Principe général :

Sur les systèmes d'exploitation Unix/Linux, l'utilisation du système passe par l'identification (login + mot de passe) d'**un utilisateur appartenant à un groupe par défaut** ayant certains droits sur les fichiers.
Tous les fichiers ont des droits définis ('rwx' / propriétaire / groupe).

Cette organisation sécurise intégralement les systèmes Unix/Linux.

root : le super-utilisateur ayant tous les droits (administrateur)
sudo : commande donnant temporairement les droits « root »

Notes :

Un groupe peut contenir zéro, un ou plusieurs utilisateurs.

Un utilisateur peut appartenir à un ou plusieurs groupes.

Toute modification du système nécessite d’être « root ».

Modification des autorisations sur les fichiers (’rwx’) :

Uniquement par le propriétaire du fichier ou l’administrateur (super-utilisateur « root » ou avec « sudo »)

Commande : chmod code fichier

Code numérique (octal) : chmod 777 fichier / chmod 000 fichier
Code symbolique : chmod ugo+x fichier / chmod ugo-x fichier
(symboles : ‘u’, ‘g’, ‘o’, ‘a’, ‘+’, ‘-’, ‘=‘, ‘r’, ‘w’, ‘x’)

Changement de propriétaire et de groupe : par l’administrateur

Propriétaire (owner) : chown nouveau-proprietaire fichier
Groupe : chgrp nouveau-groupe fichier

Notes :

« chmod », « chown » et « chgrp » s’appliquent sur les fichiers et les répertoires.

Les répertoires doivent avoir l’autorisation ’x’ (exécution) pour être accessibles.

II. Gérer les fichiers, utiliser les éditeurs

II. Gérer les fichiers, utiliser les éditeurs

1) Documentation en ligne de commande
2) Les commandes de base pour les fichiers et les répertoires
3) Les liens physiques et symboliques
4) Commandes complémentaires
5) Les différents éditeurs
6) L’éditeur vi

1) Documentation en ligne de commande

Commande « man » : manual

> Affiche les pages du manuel système.

Chaque argument donné à « man » est généralement le nom d'une commande (externe), d'un programme, d'un utilitaire ou d'une fonction.

> Ex :

man man

Affiche les informations pour l'utilisation de « man »

('q' pour quitter).

Interactivité de « man » :

Lorsqu'une page de manuel est affichée, diverses actions sont accessibles via des **raccourcis claviers** :

flèches : Navigation dans la page de manuel

'q' : Quitte

'h' : Affiche l'aide de « man »

'/' : Recherche avant (entrer le mot à rechercher + ENTREE)

'?' : Comme '/' mais recherche arrière

'n' : Va à l'occurrence suivante de la recherche

'N' : Va à l'occurrence précédente de la recherche

Notes :

Commandes internes (bash) :

help
help commande

2) Les commandes de base pour les fichiers et les répertoires

Commande « ls » : list segment

> Permet de lister un répertoire.

> Options :

-l : Permet un affichage détaillé du répertoire (permissions d'accès, le nombre de liens physiques, le nom du propriétaire et du groupe, la taille en octets, l'horodatage).

-a : Permet l'affichage des fichiers et répertoires cachés (ceux qui commencent par un '.').

> Ex :

ls -a

ls /etc/

Commande « pwd » : print working directory

> Affiche le répertoire en cours depuis la racine.

Commande « cd » : change directory

> Permet de changer de répertoire.

> Ex :

cd

Permet de revenir à /home/utilisateur (identique à **« cd ~ »**).

cd -

Permet de revenir au répertoire précédent.

cd ..

Permet de remonter au répertoire parent.

cd /

Permet de remonter à la racine.

cd /usr/bin/

Permet de se déplacer dans le répertoire précisé.

Commande « cat » : concatenate

> Affiche le contenu d’un fichier (ou de plusieurs à la suite).

> Option :

-n : Affiche les numéros de ligne.

> Ex :

cat -n monFichier

Affiche monFichier en numérotant les lignes à partir de 1.

Notes :

Créer un fichier vide : touch vide.txt

Commande « more » : plus

> Affiche un fichier page par page (ou ligne par ligne).

> Options :

-s : Regroupe les lignes vides consécutives en une seule.

-f : Ne coupe pas les lignes longues.

> Ex :

more -sf monFichier

Affiche monFichier page par page en concaténant les lignes vides sans couper les lignes longues.

Commande « less » : équivalente à « more » (affichage avant et arrière)

Commande « rm » : remove

> Permet d’effacer des fichiers.

> Options :

-f : Ne demande pas de confirmation avant d’effacer.

-r : Efface récursivement les fichiers ainsi que les répertoires.

> Ex :

rm CeFichier

Efface le fichier CeFichier.

rm -rf /tmp/LeRep

Efface le répertoire LeRep/ dans /tmp/ ainsi que tous ses fichiers sans demander de confirmation.

Commande « mkdir » : make directory

> Crée un répertoire vide.

> Options :

-p : Crée les répertoires parents s'ils n'existent pas.

> Ex :

mkdir photos

Crée le répertoire photos/.

mkdir -p photos/2005/noel

Crée le répertoire noel/ et, s'ils n'existent pas, les répertoires 2005/ et photos/.

Commande « rmdir » : remove directory

> Supprime un répertoire vide.

> Options :

-p : Supprime les répertoires parents s'ils deviennent vides.

> Ex :

rmdir LeRep

Supprime le répertoire LeRep/ s'il est vide.

> Rappel :

rm -r LeRep

Supprime le répertoire LeRep/ même s'il n'est pas vide.

Commande « cp » : copy

> Permet de copier des fichiers ou des répertoires.

> Options :

-a : Copie en gardant les droits, propriétaires, groupes, dates.

-i : Demande une confirmation avant d'écraser.

-f : Si le fichier de destination existe et ne peut être ouvert, alors le détruire et ressayer.

-r : Copie un répertoire et tout son contenu.

-u : Ne copie que les fichiers plus récents ou qui n'existent pas.

-v : permet de suivre les copies réalisées en temps réel.

> Ex :

cp monFichier sousrep/

Copie monFichier dans le répertoire sousrep/.

cp -r monRep/ ailleurs/

Copie le répertoire monRep/ vers ailleurs/ en créant le répertoire s'il n'existe pas.

Commande « mv » : move

> Permet de déplacer ou renommer des fichiers et répertoires.

> Options :

-f : Ecrase les fichiers de destination sans confirmation.

-i : Demande confirmation avant d'écraser.

-u : N'écrase pas le fichier de destination si celui-ci est plus récent.

> Ex :

mv monFichier unRep/

Déplace monFichier dans le répertoire unRep/.

mv unRep/monFichier .

Déplace monFichier du répertoire unRep/ où l'on se trouve.

mv unRep monRep

Renomme unRep en monRep.

Notes :

Mettre en pratique ces commandes.

3) Les liens physiques et symboliques

Commande « ln » : link

> Crée un lien (physique ou symbolique) vers un fichier ou un répertoire.

> Options :

aucune : Crée un lien physique (sorte de copie liée au même inode).

-s : Crée un lien symbolique (similaire au raccourci Windows).

-f : Force l'écrasement du fichier de destination s'il existe.

-d : Crée un lien sur un répertoire (uniquement en mode « sudo » / « root »).

Notes :

Un lien physique n’est possible que sur la même partition (car il pointe directement sur l’inode du fichier cible).

Un lien symbolique est possible entre deux partitions différentes (car il ne pointe pas sur l’inode du fichier cible, mais sur son nom).

> Ex :

ln -s Rep1/Rep2/Monfichier MonLien

Crée un lien symbolique MonLien de Rep1/Rep2/Monfichier dans le répertoire où l'on se trouve.

ln Monfichier unRep/AutreNom

Crée un lien physique AutreNom de Monfichier dans le répertoire unRep/.

Remarque : préférer les chemins absolus aux chemins relatifs.

Notes :

rm lien : supprime le lien et non le fichier lié.

4) Commandes complémentaires

Recherche de fichiers / répertoires : **find / -name passwd***

Recherche de textes : **grep -i "Root" /etc/passwd**

5) Les différents éditeurs

Important :

Pour l’écriture des scripts Shell, ou des sources en langage C, toujours utiliser un vrai éditeur de texte (ASCII brut), et non un logiciel de traitement de texte avec mise en forme (comme Word, OpenOffice, LibreOffice,…).

Les éditeurs de texte Unix / Linux :

ed (EDitor) : éditeur orienté ligne
emacs : éditeur orienté page
nano : éditeur simple orienté page
pico : éditeur ancêtre de nano
vi (Visual Interface) : éditeur présent dans tous les systèmes Unix (le 1er orienté page)
vim : VI aMélioré (ou VI iMproved)

Les éditeurs de texte avec GUI : gedit (GNOME), kedit (KDE), TextEdit (Mac OS X),...

6) L’éditeur vi

1er éditeur de texte plein écran, présent dans tous les Unix/Linux.

Accès en ligne de commande : vi fichier

Fonctionnement : 3 modes

Commande : mode insertion, déplacement, correction, copier,…
Insertion : mode saisie du texte (buffer)
Exécution (caractère ‘:’) : mode commandes enregistrer, quitter,…

Commandes : pages suivantes

Commande	Touche(s)	Fin
Insertion sous le curseur	`i`	`ESC`
Insertion en début de ligne	`I`	`ESC`
Ajout devant le curseur	`a`	`ESC`
Ajout en fin de ligne	`A`	`ESC`
Rajout d'une ligne sous le curseur	`o`	`ESC`
Rajout d'une ligne au-dessus du curseur	`O`	`ESC`
Déplacement vers la droite	`l` ou →	
Déplacement vers la gauche	`h` ou ←	
Déplacement vers le haut	`j` ou ↑	
Déplacement vers le bas	`k` ou ↓	
Déplacement de *4* lignes vers le bas	`4k` ou `4`↓	
Placement à la ligne *23*	`:23` puis `Return`	
Effacement d'un caractère	`x`	

Effacement de *5* caractères	`5x`	
Effacement d'une ligne	`dd`	
Effacement de *3* lignes	`3dd`	
Sauvegarde d'un fichier	`:w` puis `Return`	
Arrêt de l'édition	`:q` puis `Return`	
Sauvegarde et arrêt	`:wq` puis `Return`	
Ajout du fichier *truc*	`:r truc` puis `Return`	
Forçage d'une commande	`!`	
Copie d'un bloc de *6* lignes	`6yy`	
Collage d'un bloc au-dessous du curseur	`p`	
Collage d'un bloc au-dessus du curseur	`P`	
Recherche de la chaîne *toto*	`/toto` puis `Return`	
Suite de la recherche vers la fin	`n`	
Suite de la recherche vers le début	`N`	

Notes :

Editer un texte avec « vi » ([Echap] si bloqué).

Cas simple :

vi texte.txt
[i]
Texte
[Echap]
:wq

III. Comprendre et utiliser le Shell

III. Comprendre et utiliser le Shell

1) Notion de Shell
2) Les variables
3) L’expansion/substitution de commandes
4) Les alias
5) Les filtres
6) Les redirections
7) Notion de scripts Shell

1) Notion de Shell

Le Shell :

Programme **interpréteur** de commandes et de scripts.
Interface entre l'utilisateur et le système d'exploitation (terminal virtuel ou physique).

Différentes versions avec spécificités : sh, bash, ksh, csh,…

A l'utilisation (commandes et scripts), toujours connaître le Shell utilisé (car risques d'incompatibilités).

2) Les variables

Variables du Shell et variables d'environnement :

Zones mémoire réservées et nommées stockant des données.
Perdues (vidées) quand la session du Shell est fermée/quittée.

Variables du Shell : utilisées que par le Shell courant.
Variables d'environnement (partagées) : transmises aux Shells fils, grâce à la commande « export ».

Création d'une variable : valeur1=10 ; repertoire="/tmp"

Visualisation des variables définies :

set (toutes) ; env (uniquement d'environnement/exportées)

Evaluer une variable : obtenir son contenu (précédée de '$')
echo $valeur1 ; echo $repertoire ; echo $PATH

Modifier le contenu d'une variable :
Réaffecter : valeur1=12
Compléter : repertoire=$repertoire/script => /tmp/script

Exemple : PATH=$PATH:/home/utilisateur

Rendre une variable exportable : héritée par les Shells fils (env.)

Export : export valeur1 (fin d'export : export -n valeur1)

Suppression de variables : unset valeur1 repertoire
Variable en lecture seule (constante) : readonly valeur1

Notes :

« bash » permet de lancer un nouveau Shell (fils) depuis le Shell courant.

« exit » permet de le quitter et de retourner au Shell appelant (père).

3) L’expansion/substitution de commandes

Rôle :

Récupérer le résultat d’une commande dans une variable ou pour une autre commande.

Syntaxe : « variable=`commande` » « variable=$(commande) »
Exemples :

```
aujourdhui=`date` ; echo $aujourdhui
compteur=0
compteur=`expr $compteur + 1`
echo $compteur
echo "On est le "$(date)
```

Notes :

Question : que vaut la variable « aujourdhui » ?

aujourdhui=date
echo $aujourdhui

4) Les alias

Rôle :

La commande « alias » permet, comme son nom l'indique, de **créer un alias ("nouvelle commande") sur une commande**, ou de fournir un alias pour la même commande avec des options supplémentaires.

Syntaxe : alias cde2='cde1 -options arguments'
Exemples :
alias dir='ls'
dir
alias ll='ls -l'
ll

Compléments :

« alias ll » rappelle la définition de l’alias « ll ».

La commande « alias » seule liste tous les alias définis en cours.

Un alias défini en cours de Shell sera perdu à la fermeture de celui-ci.

Les alias sont habituellement définis dans le fichier de démarrage du Shell (« ~/.profile » ou « ~/.bashrc » pour bash).

« unalias ll » (ou « alias ll= ») supprime l’alias « ll », mais seulement pour le Shell courant. Il faut aussi supprimer sa définition dans le fichier idoine (si ajouté).

5) Les filtres

Rôle :

Renvoyer la sortie (le résultat affiché) d'une 1ère commande vers l'entrée d'une 2ème commande, à l'aide du **symbole '|' (pipe)**.

La 2ème commande exploite le résultat (affiché) de la 1ère pour en filtrer les données.

Syntaxe : cde1 | cde2

Quelques commandes orientées filtres :

more, sort, grep, wc, cut, tr, head, tail,…

Exemples :

ls -la | more
ls | wc
set | sort
set | grep TERM
set | sort | more
cat fichier | sort | more

6) Les redirections

Redirection en sortie :

Au lieu d'afficher le résultat d'une commande à l'écran (sortie standard), le **rediriger dans un fichier** à l'aide des symboles '>' (fichier créé ou écrasé) ou '>>' (fichier créé ou complété).

Syntaxe : « commande1 > fichier » « commande2 >> fichier »
Exemples :
ls -la > fichier1
set > fichier1
cat > fichier2 (CTRL+D pour sauver)
cat >> fichier2

Redirection en entrée :

Au lieu d'utiliser la saisie au clavier pour les paramètres d'une commande (entrée standard), les **récupérer depuis un fichier** à l'aide du symbole '<'.

Syntaxe : « commande < fichier »

Exemples :

wc < fichier1

grep TERM < fichier1

Combinaison entrée/sortie :

wc < fichier1 > fichier2

7) Notion de scripts Shell

Définition :

Programmes Shell **écrits en fichier texte** (ASCII).

Avec fonctions, commandes (internes/externes), mots clés, #,…

Une commande script peut être un autre script Shell.

1ère ligne (shebang) : #!/bin/bash (pour utiliser le Shell « bash »).

Création d'un script :

Analyse préalable du besoin + Editeur de texte (vi ou autres) :

```
if [ $# -eq 0 ]
        then
                echo "Aucun paramètre transmis"
fi
```

Structure d’un script :
Linéaire et séquentielle, instructions exécutées en série (par défaut).

Exécution d’un script : interprétation (pas de compilation)

Rendre le script exécutable : **chmod ugo+x monscript.sh**

Exécution en fonction du Shell précisé (interpréteur) :

« bash monscript.sh » « ksh monscript.sh » « sh monscript.sh » …

Ou simplement le nom (avec interprétation du shebang) si script présent dans un répertoire de $PATH, sinon préciser le chemin relatif ou absolu : **./monscript.sh** (répertoire courant)

Suppression d’un script : rm monscript.sh

IV. Le fonctionnement multitâche et les processus

IV. Le fonctionnement multitâche et les processus

1) Exécution en arrière plan
2) Outils pour le background
3) Signaux et suppression d’un processus

1) Exécution en arrière plan

Rappel :

Unix/Linux est un système multitâches, c'est-à-dire qu'il peut exécuter plusieurs programmes "à la fois".

Un processus (process en anglais) est une instance d'un programme en train de s'exécuter (ou suspendu), **une tâche en mémoire**. Le Shell crée un nouveau processus pour exécuter chaque commande / programme.

En monotâche, si on lance une commande qui prend beaucoup de temps (comme un calcul), **on peut l'interrompre par CTRL+C**. Ceci interrompt définitivement la commande.

Suspendre et reprendre l'exécution d'une commande :

Suspendre un programme : **CTRL+Z => [n] : n° du job (Shell en cours)**

Ramener le programme en 1er plan : **fg %1 (si n° du job = [1])**

Tester avec : « sleep 3600 »

Exécution en tâche de fond : en arrière plan (background)

Faire suivre la commande **d'un espace et d'un '&'**

=> [n] : n° du job + PID : n° du processus (unique)

L'invite / prompt se réaffiche immédiatement (pour une autre commande)

Tester avec : « sleep 3600 & »

2) Outils pour le background

jobs : liste des programmes + n° de leur job (Shell en cours)

fg %n : ramène la tâche en premier plan (exécution foreground)

bg %n : ramène la tâche en arrière plan (exécution background)

Tester avec : « sleep 3600 » et CTRL+Z

Commande « ps » : processes snapshot

> Affiche les processus en cours avec leur PID.

> Options :

-u : Affiche les processus de l'utilisateur qui exécute la commande.

-au : Affiche les processus de tous les utilisateurs.

-aux : Affiche l'intégralité des processus du système. Variante « ps -A ».

-faux : Affiche tous les processus du système en les regroupant par enchaînement d'exécution.

> Ex :

ps -u

Tous les processus de l'utilisateur courant.

ps -aux

Tous les processus en cours.

Commande « top » : top CPU

> Affiche en temps réel la charge CPU et mémoire par processus.

> Options :

-u : Affiche les processus pour un utilisateur donné.

> Ex : top -u root

(’q’ pour quitter)

Notes :

« pstree » affiche sous forme d’arborescence la liste des processus avec leur hiérarchie « père / fils ».

3) Signaux et suppression d’un processus

Les signaux Unix / Linux :

Un signal est un message envoyé à un processus. Le corps du message est très simple : **un entier indiquant le type du signal**.
Tout processus peut envoyer un signal à un autre processus, à partir du moment où il **connaît son PID**.

Taper CTRL+Z au clavier pour suspendre un processus lui envoie un signal de suspension, de même que « fg » et « bg » envoient un signal de reprise d’exécution.
« kill », pour tuer un programme récalcitrant, est en fait destiné à **envoyer tout type de signal**.

Commande « kill » / « killall » : tuer par PID / tuer par nom (tous)

> Permet d'envoyer un signal à un processus : « kill » ne comprend que les PID (Process IDentifier, n° d'ordre du processus), « killall » lui comprend le nom du processus.

> Options :

-s : Indique quel signal à envoyer au processus. Le signal peut être identifié soit par son nom (exemple : SIGTERM), soit par son numéro (exemple : 9). Remarque : « -s 9 » est équivalent à « -9 ».

-l : Affiche la liste des signaux connus (n° et noms).

> Les signaux les plus courants sont :

TERM, signal 15 : Le signal « Terminate » indique à un processus qu'il doit s'arrêter (signal par défaut).

KILL, signal 9 : Le signal « Kill » indique au système qu'il doit arrêter un processus sans condition.

HUP, signal 1 : Signal de fin d'exécution où le processus doit relire son fichier de configuration.

Exemples :

kill -15 14774 : Envoie le signal 15, ou TERM, au processus ayant le numéro 14774, ce qui a pour effet de terminer proprement le processus.

kill -9 7804 : Envoie le signal 9, ou KILL, au processus ayant le numéro 7804, ce qui a pour effet de tuer le processus (sans condition).

killall -TERM firefox : Envoie le signal TERM, ou 15, au processus firefox, ce qui a pour effet de le fermer.

Il est conseillé de lancer des signaux de faible importance avant de lancer des signaux forts. En pratique, tester dans l'ordre chacune de ces commandes :

kill *pid* : envoie le signal 15 (TERM)

kill -KILL *pid* : envoie le signal 9 (KILL)

V. Unix/Linux en réseau local

V. Unix/Linux en réseau local

1) Notions sur TCP/IP
2) Commandes distantes sécurisées (ssh)

1) Notions sur TCP/IP

Protocole : description des formats de messages et règles selon lesquelles 2 ordinateurs échangent des données.

TCP/IP : Transmission Control Protocol/Internet Protocol. Protocole utilisé sur le réseau Internet pour transmettre des données entre 2 machines.

Protocole de transport : TCP prend à sa charge l'ouverture et le contrôle de la liaison entre 2 ordinateurs.
Protocole d'adressage : IP assure le routage des paquets de données à l'aide d'adresses.

Adresse IP : numéro (ex : 192.168.10.1) qui identifie tout matériel informatique (ordinateur, routeur, imprimante,...) connecté à un réseau informatique utilisant l'Internet Protocol.
Versions actuelles : IPv4 (4 octets) & IPv6 (16 octets).

IP dynamique et IP fixe : l'adresse IP d'un ordinateur lui est - généralement - automatiquement assignée au démarrage par un serveur et le protocole DHCP (Dynamic Host Configuration Protocol). Il est également possible de fixer manuellement l'adresse IP au niveau de la configuration du système.

Port : en complément de l'IP, numéro qui indique l'application à laquelle les données sont destinées (ex : 21 = FTP, 80 = HTTP).

Notes :

Le protocole réseau natif d’Unix / Linux est TCP/IP.

Liste des services avec leur numéro de port : « more /etc/services »

Principales commandes réseau Unix/Linux :

Liste des interfaces réseau détectées (toutes) : **« ifconfig »**

Informations sur le Wi-Fi (wlan) : **« iwconfig »**

Informations sur le routage : **« route -n »**

Tester l'adresse locale (127.0.0.1) et autres IP :

« ping localhost »
« ping 192.168.10.1 »

Voir les utilisateurs connectés : **« who »**

2) Commandes distantes sécurisées (ssh)

« ssh » est à la fois une commande Unix/Linux et un **protocole de communication sécurisée** (port 22). Il permet de se connecter à un ordinateur distant afin d'obtenir un Shell avec ligne de commande (terminal), et possède un protocole de transfert de fichiers complet.

Syntaxe :

« ssh machinedistante -l nom »
« ssh nom@machinedistante »

Puis saisie du password (« exit » pour sortir).

Notes :

ssh = secure shell.

Copier des fichiers via SSH :

Pour copier un fichier à partir d'un ordinateur sur un autre avec SSH, utiliser la commande **« scp »** :

scp <fichier> <user>@<ipaddress>:<destination>
scp -6 <element> <nom>@[addresse ipv6]:<destination>

Exemple :

« scp fichier.txt utilisateur@192.168.1.103:/home/utilisateur »
« scp utilisateur@192.168.1.103:/home/utilisateur/fichier.txt . »

Et aussi « sftp » (FTP sécurisé) : « sftp utilisateur@192.168.1.103 »

Notes :

scp = secure copy.

Utiliser l’option « -r » pour copier des répertoires.

sftp = secure file transfer protocol/program.

Démarrer / Arrêter / Relancer le service SSH (et état / statut) :

sudo service [ssh|sshd] start

sudo service [ssh|sshd] stop

sudo service [ssh|sshd] restart

sudo service [ssh|sshd] status