1. Généralités sur Unix

Un **système d'exploitation** est un ensemble de programmes qui contrôlent et organisent l'usage des ressources d'un ordinateur.

Quelques **caractéristiques** essentielles du système d'exploitation Unix :

- système d'exploitation "ouvert", non propriétaire
- en grande partie (>95%) écrit en langage évolué (langage C)
- disponible sur toutes les plateformes, y compris PC (Linux, FreeBSD...)
- multi-tâche et multi-utilisateur
- interactif, temps partagé, mais offrant aussi des extensions pour le "temps réel"
- multi-processeur symétrique, multi-threaded
- gestion de mémoire virtuelle, mémoire protégée
- facilement extensible et modifiable (ajout de commandes...)
- "case sensitive", c'est à dire qu'il distingue les majuscules des minuscules (tant au niveau des commandes et options que des noms de fichiers) contrairement à VMS ou MS-DOS; ex: la commande "`ls -l`" n'a pas même effet que "`ls -L`")

Unix se **compose** principalement de :

- noyau d'exécution (kernel) (partie du système chargée en mémoire vive s'occupant de la gestion des processus, de la mémoire, des entrées/sorties)
- système de gestion de fichiers hiérarchisé
- interpréteurs de commandes (shells)
- commandes et utilitaires (programmes exécutables externes au shell)
- gestionnaire de processus (multi-tâches, multi-utilisateurs)
- services de communication (basés sur le protocole internet TCP/IP)

Figure 1 : Architecture Unix

S'ajoute à Unix, sur une station de travail, un **environnement graphique** (système de fenêtrage) basé sur **X-window** composé de :

- serveur X-window
- gestionnaire de fenêtre (p.ex. CDE Window Manager)
- applications graphique de base (gestionnaire de fichiers, émulateur terminal, outils de bureau...) (p.ex. outils CDE)

Le **système de fichier** est une organisation logique d'un ensemble de disques physiques. L'ensemble des **répertoires** (**directoires**, ou **dossiers** dans la terminologie PC) forme une structure hiérarchique. Le système de gestion de fichiers Unix est arborescent et n'a qu'une seule **racine** : "`/`" (root directory, sommet de la hiérarchie) par oposition à Windows p.ex. qui en a plusieurs (A:, C: etc...)

Unix contient un certain nombre de répertoires-système spéciaux (`/usr, /etc, /dev, /var`...). Dans l'environnement Unix ENAC-SSIE, tous les utilisateurs se trouvent sous l'arborescence `/USERS/`*`nom_utilisateur`*

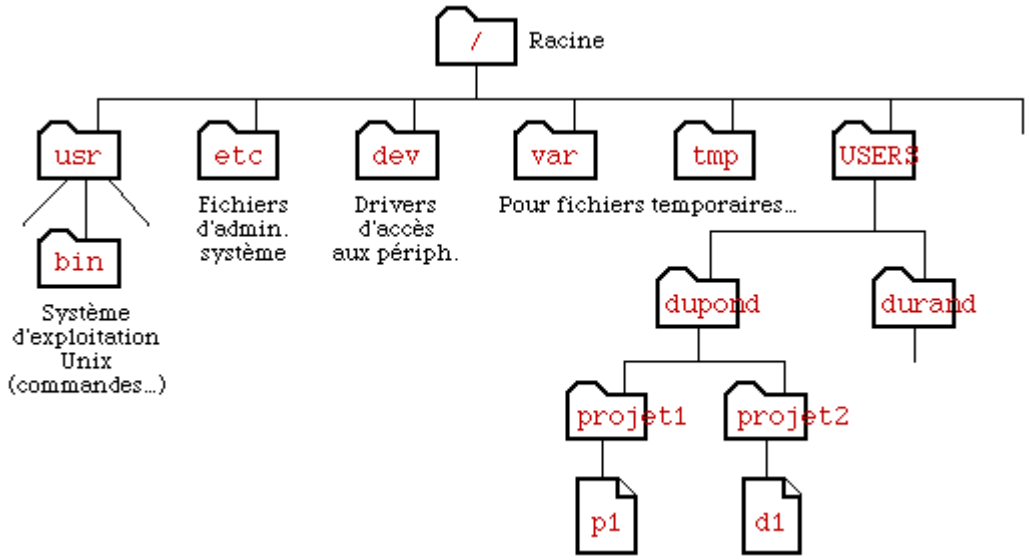

Figure 2 : Système de fichier Unix

Un **fichier** est désigné par son **nom** et par le **chemin d'accès** (**path**) menant, depuis la racine, jusqu'à son emplacement dans la hiérarchie de répertoires. Le nom de fichier est composé d'une suite de 1 à 255 caractères (seulement 14 sur les anciens Unix System V). Dans la pratique, on se limite généralement aux caractères : a-z, A-Z, 0-9, "_", "-" et "."

Toutes les commandes et applications cherchent et déposent par défaut leurs fichiers dans le **répertoire courant** (défini par l'utilisateur). Lorsque l'on veut accéder à un fichier qui n'est pas dans le répertoire courant, il faut faire précéder son nom du **path** qui est la séquence des noms de répertoires conduisant jusqu'à ce fichier, chaque nom de répertoire étant séparé du suivant par le caractère "`/`".

Quelques **caractères spéciaux** dans nom de fichier et path, interprétés par le shell :

- `/` = racine du système de fichier (root directory) ainsi que séparateur de noms de répertoires
- `..` = répertoire situé au-dessus du répertoire courant
- `.` = répertoire courant
- ~ = répertoire principal de l'utilisateur (**home-directory**) (en C-shell)
- métacaractères (caractères de substitution, identiques en C-shell et en Bourne-shell) :
  - `*` = remplace 0, 1 ou plusieurs caractères (ex: `chap*.doc` désignerait p.ex. fichiers `chap1.doc` et `chap12.doc`)
  - `?` = remplace 1 caractère quelconque

Un **path** peut être "absolu" ou "relatif" :

- Le path **absolu** est exprimé à partir de la racine "/".
  Ex : `/USERS/dupond/projet1`
- Le path **relatif** est exprimé par rapport au répertoire courant.
  Ex 1 : si on est dans `dupond`, le path `projet1` descend dans projet1,
  puis `../projet2` se déplace horizontalement de projet1 dans projet2
  Ex 2 : `../../..` = trois répertoires plus haut que le répertoire courant

Unix étant **multi-utilisateur**, le système de fichier maintient également des informations indiquant :

- propriétaire du fichier (utilisateur l'ayant créé et groupe) (visible avec commande `ls -l`)
- droits d'accès des autres utilisateurs (groupe, autres) sur ce fichier (lire, écrire, exécuter) (voir commande `chmod`)
- date et heure de modification et de dernier accès au fichier

---

2. Commandes de base

2.1 Conventions de notation dans ce document

En police de caractère `machine à écrire` (police "Courier" à espacement fixe)
    Texte qui doit être entré tel quel au clavier (ex: "`ls`")
En *`itallique`*
    Vous devez substituer vous-même l'information indiquée
    (ex: "`cp` *`fich_source fich_destination`*")
Entre accolades { }
    Eléments optionnels (ex: "`ls {-al}` {*`path`*}")
Entre < >
    Touche du clavier ou combinaison de touches (ex: `<CR>`, `<ctrl-C>`)
Entre [ ]
    Origine, au sens étymologique, des mnémoniques de commandes et des options
    (ex. pour la commande "`rm`" : [*remove*])
Texte souligné
    Liens hyper-texte vers d'autres documents (exploitables depuis browser WWW)

2.2 Prompt

Par l'affichage de son "prompt", le shell "invite" l'utilisateur à entrer une commande. Dans l'environnement ENAC-SSIE, le prompt par défaut est du type :
`[NoCommande]username@machine:/répertoire_courant>`

La commande est interprétée par le shell puis exécutée lorsque l'utilisateur presse la touche `<CR>`.

2.3 Forme d'une commande

prompt> *verbe {-option(s)} {paramètre(s)}* <CR>

- le *verbe* (action) est un mnémonique très court, en principe toujours en caractères minuscules
- un ou plusieurs caractères " " (espace) ou <TAB> sont introduits comme séparateurs entre *verbe*, les *options* et *paramètre(s)*
- le caractère "-" introduit les *options* de la commande. Généralement, les options peuvent être regroupées (ex : "ls -alL" est équivalent à "ls -a -l -L")
- certaines options peuvent avoir des arguments, parfois collés à l'option (ex: "f77 -l*librairie* ...") parfois séparés par un espace (ex: "pc -o *fich_exécutable* ...")

2.4 Touches et caractères spéciaux

- <CR> = terminaison et lancement d'une commande
- <DEL> = efface le dernier caractère
- <ctrl-C> = interrompt la commande (processus) en cours d'exécution
- <ctrl-Z> = suspend le travail en cours d'exécution (le relancer avec commande fg)

Sur clavier Sun "Type 5 Suisse-romand", pour accéder aux caractères spéciaux notés au bas à droite de certaines touches, utiliser simultanément la touche **<AltGraph>** (située deux touches en-dessous de la touche <CR>). Exemple : pour obtenir le caractère "|" (pipe), frapper <AltGraph-1>

2.5 Commandes principales

- Si vous connaissez déjà l'un des systèmes d'exploitation DOS ou VMS, vous trouvez en cliquant ici un tableau de correspondance entre les principales commandes Unix, VMS et DOS
- Seules les commandes élémentaires et leurs options de base sont présentées ci-dessous

Mot de passe

passwd
: Changement de son mot de passe. Lorsque le système le demande, introduire "à l'aveugle" son ancien mot de passe puis le nouveau mot de passe de son choix (au min. 6 caractères dont 1 chiffre)

Manuel Unix en ligne

man *commande*
: Demande d'aide sur la *commande* spécifiée [*manual*]. (Pour davantage d'info. cliquer ici)

Pour des informations encore plus détaillées, consulter l'AnswerBook Sun (à partir du Root-Menu ou du Front Panel CDE)

Alias

alias *nom_alias 'commande {options} {paramètres}'*
: Crée un alias (raccourci de commande) *nom_alias* pour la *commande* spécifiée (avec ses éventuels *options* et *paramètres*)

Passage d'arguments dans un alias (C-shell) :

- `\!*` = substitution de tous les arguments
- `\!^` = substitution du premier argument
- `\!$` = substitution du dernier argument
- `\!:`*n* = substitution du *n*-ème argument

Exemple : si on a défini : `alias psg 'ps -ef | grep \!* | more'`
on pourra passer la commande : `psg toto`
qui est équivalente à : `ps -ef | grep toto | more'`

Répertoires

`ls {`*fichier(s) ou répertoire(s)*`}`
: Liste le contenu du répertoire courant {ou le(s) *fichier(s)* ou *répertoire(s)* spécifiés} [*list files*]
`-a` = tous les fichiers, y compris hidden-files (fichiers dont le nom commence par ".")
`-l` = listage long : droits d'accès (voir commande `chmod`), liens, propriétaire, taille [bytes], date et heure de modification
`-R` = liste en parcourant récursivement tous les sous-répertoires

`cd {`*répertoire*`}`
: Change de répertoire courant [*change directory*]. Si on ne donne pas de nom de répertoire, renvoie dans le répertoire principal de l'utilisateur (home)

`pwd`
: Affiche le chemin d'accès du répertoire courant [*path of working directory*]

`mkdir` *répertoire*
: Crée *répertoire* de nom spécifié [*make directory*]

`rmdir` *répertoire*
: Détruit le *répertoire* spécifié (dont le contenu doit avoir préalablement été détruit) [*remove directory*]

Visualisation de fichiers

`cat` *fichier*
: Affiche le contenu du *fichier* texte spécifié

`cat` *fichier1 fichier2 fichier3 > fichierf*
: Concatène fichiers 1, 2 et 3 dans *fichierf* [*catenate*]

`more` *fichier*
: Affichage contrôlé du *fichier* texte spécifié. L'avancement dans le fichier est contrôlé par les commandes :
`<SPACE>` = page suivante
`<CR>` = ligne suivante
/chaîne = cherche la chaîne de caractères spécifiée
`n` = cherche la prochaine occurence de cette chaîne
`q` = termine affichage (sortie de more)

`head {-`*n*`}` *fichier*
: Affiche les 10 {ou *n*} premières lignes de *fichier*

`tail {-`*n*`}` *fichier*
: Affiche les 10 {ou *n*} dernières lignes de *fichier*
+*n* = à partir de la *n*-ème ligne jusqu'à la fin du fichier

Manipulation de fichiers

`cp` *fich_source fich_dest*
: Copie *fichier source* sur *fichier destination* [*copy*]

`cp` *fichier(s) répertoire*

Copie *fichier(s)* dans *répertoire*

`mv fich_source fich_dest`

Renomme fichier

`mv fichier(s) répertoire`

Déplace *fichier(s)* dans *répertoire* [*move*]

`mv rép_source rép_dest`

Renomme répertoire ou le déplace dans autre répertoire

`rm {-i} fichier(s)`

Détruit *fichier(s)* [*remove*]

Avec `-i`, demande confirmation à l'utilisateur (option activée par défaut en ENAC-SSIE)

`chmod`

Changement des droits d'accès d'un fichier ou répertoire [*change mode*]. (Pour plus d'info. sur cette commande, cliquer ici)

Edition de fichiers-texte

`vi fichier`

Editeur de base, commun à tous les Unix, mais pas très convivial [*visual editor*]. (Pour plus d'info. cliquer ici)

`emacs fichier`

`xemacs fichier`

Editeur de la Free software fundation (GNU), en version terminal resp. X-window

`dtpad fichier`

Editeur de base de l'environnement CDE

Utilisation de l'espace disque

`du {répertoire}`

Affiche l'espace-disque utilisé par toute l'arborescence courante {ou par celle spécifiée} [*directory usage*]

`-a` = affiche non seulement les répertoires mais aussi les fichiers

`-k` = affiche les tailles en Kbyte (exprimées sinon en blocs de 512 bytes)

Passée dans votre home, la commande "`du -k`" vous indique donc en KB la place disque totale que vous occupez

`quota -v`

Affiche l'usage de l'espace-disque occupé par l'utilisateur (maximum autorisé et espace utilisé) sur chaque file-system

Comparaisons, recherches, tris

`diff fichier1 fichier2`

Compare ligne par ligne le contenu des fichiers texte *fichier1* et *fichier2*, et affiche les différences

`grep 'motif' fichier(s)`

Affiche les lignes de *fichier(s)* contenant chaîne de caractères satisfaisant à expression régulière *motif* [*global regular expression print*]. Voir aussi commande `egrep` plus riche [*extended grep*] et `fgrep`

`-i` = ne distingue pas majuscules et minuscules

`-v` = affiche toutes les lignes ne satisfaisant pas la condition

`sort fichier_entree > fichier_sortie`

Trie alphabétiquement (selon table ASCII) ligne par ligne le contenu de *fichier_entree* et envoie résultat sur *fichier_sortie*

`-n` = trie numériquement plutôt qu'alphabétiquement ("10" viendra après "2")

`-r` = trie dans ordre inverse
`-f` = ne distingue pas majuscules et minuscules

Impression

`lp {-d` *`imprimante`*`} `*`fichier(s)`*
Envoie le(s) *fichier(s)* sur la file d'attente de l'*imprimante* spécifiée [*line printer*]
Si aucune *imprimante* n'est spécifiée, utilise celle définie par l'utilisateur par la variable d'environnement `LPDEST` (ou l'imprimante par défaut définie par l'administrateur)

`lpstat {`*`imprimante`*`}`
Indique l'*Id* (sous la forme *imprimante-No*) et le statut des requêtes d'impression en attente sur toutes les imprimantes (ou l'imprimante spécifiée) [*line printer status*]

`cancel` *`Id(s)`*
`cancel -u` *`username`*
Avorte la(les) job(s) d'impression spécifié(s) par *Id(s)*,
ou avorte tous les jobs d'impression soumis par l'utilisateur *username*

Compression

`compress` *`fichier`*
Compresse *fichier* (réduction de taille de l'ordre de 50%). Le résultat est un fichier de nom *`fichier`*`.Z`, et le *fichier* original disparaît du répertoire

`uncompress` *`fichier`*`{.Z}`
Décompresse fichier *fichier*.Z compressé avec `compress`. Le résultat est un fichier de nom *fichier*, et le *fichier*.Z disparaît du répertoire

Les commandes suivantes seront utilisées en fonction des extensions rencontrées :

`.gz`
utiliser commande "`gunzip`"

`.zip`
utiliser commande "`unzip`"

`.uu`
utiliser commande "`uudecode`"

`.tar`
utiliser commande "`tar`"

Processus, jobs

`ps -ef {| grep` *`username`*`}`
Affiche la liste de tous les processus {ou seulement ceux de l'utilisateur *username*}, leur *PID*, leur état, le temps CPU utilisé, la commande exécutée... [*process status*]

`kill -9` *`PID`*
Tue processus de *PID* spécifié

`commande` `&`
Exécute *commande* en arrière-plan (background). Elle s'exécute donc de façon "détachée"du shell courant. Le job en question peut survivre au shell et à la session à partir desquels il a été lancé
(Identique à la séquence : "*`commande`* `<CR> <ctrl-Z> bg`")

`jobs`
Liste les jobs du shell courant (suspendus ou s'exécutant en arrière-fond) avec leurs *jobID* (numérotés dans le cadre du shell auquel ils sont rattachés) [*jobs status*]

`kill -9 %`*`jobID`*
Tue le job de *jobID* spécifié

Commandes diverses

`date`

Affiche la date et l'heure courante

`who`

Affiche la liste des utilisateurs connectés sur la machine courante

`echo {"}chaîne de caractères{"}`
`echo $`*`variable`*

Affiche *chaîne de caractères* ou contenu de *variable* sur la sortie standard

`-n` = sans faire de saut de ligne à la fin de l'affichage

2.6 Redirection et tube

`commande` `>{>}{!}` *`fichier_sortie`*

Redirige la sortie de la *commande* ("sortie standard") vers *fichier_sortie*

`>>` = en mode append si *fichier* préexiste

`>!` = écrase *fichier* s'il préexiste

`commande1` `|` *`commande2`*

La sortie de *commande1* est envoyée en entrée dans la *commande2*