ABC d'UNIX

J-C. Soetens
jean-christophe.soetens@u-bordeaux.fr

Contents

1 Présentation d'UNIX

Unix est né en 1969 aux laboratoires Bell (filiale d'ATT) sous l'impulsion de Ken Thompson et Dennis Ritchie pour répondre à un usage interne. Leur objectif était de développer un système d'exploitation interactif pour des petites machines dotées de possibilités comparables à celles des grands systèmes. Unix sera ensuite réécrit en langage C (inventé par Dennis Ritchie) à partir de 1973, ce qui le rendra par conséquent portable sur un grand nombre d'ordinateurs.

UNIX est un système d'exploitation multi-tâches et multi-utilisateurs. En tant que système d'exploitation, son principal rôle est d'assurer une répartition optimale des ressources (mémoire, processeur(s), disques...) entre les différentes tâches des différents utilisateurs.

Les fonctions principales d'UNIX sont :

- *Connexion et déconnexion* : une fois que l'administrateur du système a enregistré un utilisateur, le système se charge de verifier l'identité de l'utilisateur lorsque celui ci désire se connecter. Un interpréteur de commandes (shell) permettant un dialogue utilisateur/système est alors automatiquement activé.
- *Gestion des ressources* de l'installation et partage de ces ressources entre les utilisateurs et les différentes tâches.
- *Gestion des données* : celle-ci consiste en l'organisation, la maintenance et l'accès aux unités de stockage (mémoire, disques durs, bandes magnétiques...)
- *Communication entre utilisateurs* : c'est par exemple le courrier électronique ou les transferts de fichiers.
- *Environnement de programmation* : ce sont les compilateurs (C, C++, Fortran...), éditeurs de textes, outils d'aide à la programmation (débogueurs...).

1.1 Déroulement d'une session de travail

La *session de travail* est l'intervalle de temps compris entre la connexion et la deconnexion de l'utilisateur. A la connexion, le système va s'assurer de l'identité de l'utilisateur et si celui ci est reconnu, un protocole de dialogue (un shell) va automatiquement s'établir. La deconnexion consiste à indiquer au système d'exploitation que la session de travail est terminée.

1.2 Procédure de connexion

C'est durant cette étape que l'utilisateur va devoir s'identifier auprès du système. Cette identification a lieu de la façon suivante :

- Affichage du message **login:** après lequel il faut rentrer son nom d'utilisateur (user id ou uid).
- Affichage du message **Password:** après lequel il faut rentrer son mot de passe. Celui-ci n'est pas affiché pendant la frappe pour éviter que quelqu'un d'autre puisse l'apercevoir.

Après la connexion, différents messages en provenance de l'administration du système (mots du jour, présence de courrier dans la boîte à lettres, etc...) sont affichés. L'utilisateur est effectivement prêt à travailler quand il reçoit l'invite du système consistant en un marqueur en début de ligne (le prompt). Ce marqueur est variable selon les machines (ex : **$** ou **nom_machine**[**numero**] > et peut être redéfinit pas l'utilisateur.

1.3 Changement de mot de passe

A la première connexion ou par sécurité, l'utilisateur peut avoir à modifier son mot de passe. Ce changement s'effectue avec la commande **passwd**. Le changement de mot de passe s'effectue en entrant d'abord le mot de passe courant puis deux fois le nouveau mot de passe. Exemple d'utilisation par l'utilisateur "gromit" :
$ passwd
Changing password for "gromit"
gromit's Old password:
gromit's New password:
Enter the new password again:
Si le nouveau mot de passe est rejeté, la raison peut être que la seconde entrée du nouveau mot de passe (pour confirmation) était différente de la première ou, éventuellement, que le nouveau mot de passe ne répond pas à des exigences de sécurité.

1.4 Procédure de déconnexion

Celle-ci dépend du type de session qui a été ouverte. Si un environnement graphique (tel que celui créé par X Windows, l'environnement multi-fenêtrage) est en place, il existe généralement un menu "logout" (ou "exit") qui permet de quitter cet environnement et ainsi de terminer la session de travail. Parfois aussi, ce menu ne permet simplement que de quitter l'environnement graphique. Il faut ensuite procéder à l'étape de déconnexion ci-dessous. En l'absence d'un environnement graphique, une simple commande telle que **logout** ou **exit** suffit pour terminer la session de travail.

2 Interpréteur de commandes

Une fois connecté, l'utilisateur peut soumettre au système des commandes. *Toute commande entrée sera interprétée par l'interpréteur de commandes (ou shell).* Le terme shell (coquille) a été choisit pour exprimer l'idée d'une interface enrobant le noyau du système et qui établit une communication entre ce noyau et l'utilisateur.

Il existe de nombreux interpréteurs de commandes sous UNIX. Les principaux (que l'on retrouve sur la plupart des systèmes) sont le Bourne-shell (sh), le C-shell (csh), le Korn Shell (ksh) et le T-CShell (tcsh). Le choix de l'interpréteur activé à la connexion est fait par l'administrateur du système à l'enregistrement de l'utilisateur et dans la plupart des cas il s'agit du C-shell ou du Korn-shell.

2.1 Syntaxe des commandes

nom_commande [`options`] [*arguments*]

- le caractère séparateur entre les différents éléments de la commande est le blanc (touche ESPACE du clavier).
- les options commencent habituellement par le caractère – (signe moins) suivi d'une ou plusieurs lettres-clés. Ces options vont modifier le comportement de la commande. Les crochets autour des arguments et des options signifient que ceux-ci sont optionnels.
- les arguments spécifient les objets (fichiers ou variables) sur lesquels la commande va s'appliquer.

Pendant la saisie de la commande, l'utilisateur peut corriger sa frappe à l'aide des touches DELETE ou BACKSPACE. Lorsque l'utilisateur a terminé la saisie, il soumet la commande au système en appuyant sur la touche ENTER. Certains shells possèdent des mécanismes de *rappel de commandes* qui permettent de réutiliser (telle quelle ou après modification) une commande précédement utilisée.

2.2 Commandes élémentaires

ls permet d'obtenir la liste des fichiers du répertoire courant.
ls `-l` `l` comme long, donne tous les attributs des fichiers.
ls `-la` `a` comme all, liste aussi les fichiers commençant par le caractère "."

cat *f1* affiche à l'écran le contenu du fichier f1.
cp *f1 f2* copie du fichier *f1* dans le fichier f2 (**co**p**y**).
mv *f1 f2* renomme le fichier f1 en f2 (**m**o**v**e).
rm *f1* détruit le fichier f1 (remote).

Remarques:

- Tous les shells font la distinction entre les lettres minuscules et majuscules pour les commandes et les noms de fichiers (contrairement au défunt MS-DOS).
- On dispose sous UNIX d'une aide en ligne permettant d'accéder aux règles d'utilisation et aux fonctionnalités d'une commande. Pour cela, il suffit de faire la commande : **man** *nom_commande*.

2.3 Caractères spéciaux du shell

Lors de la saisie d'une commande (**nom_commande** [`options`] [*arguments*]), tout ou une partie de l'*argument* peut être désigné de façon subtile à l'aide de caractères spéciaux :

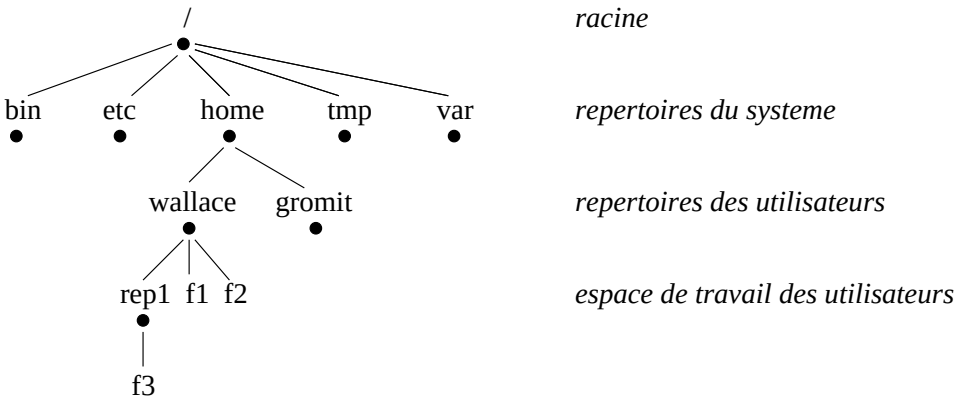
? désigne un caractère quelconque.
∗ désigne une suite (qui peut être vide) de caractères .
[...] désigne un caractère de la liste.
[^...] désigne un caractère n'appartenant pas à la liste.

Exemple: soit un répertoire courant tel que :
ls donne `fich1.bin fich1.txt fich2.txt fich10.txt fichier.txt readme zzz`
alors
ls *fich1∗* donne `fich1.bin fich1.txt fich10.txt`
ls *fich∗.txt* donne `fich1.txt fich2.txt fich10.txt fichier.txt`
ls *fich*[*0-9*]*∗.txt* donne `fich1.txt fich2.txt fich10.txt`
ls *???* donne zzz

3 Organisation arborescente des fichiers d'UNIX

L'unité d'information gérée par le système est le fichier. Selon leur utilisation, les fichiers sont appelés répertoires ou fichiers tout court. Un répertoire est un catalogue de fichiers contenant leurs caractéristiques comme les droits d'accès, la taille, la date de création... *L'ossature du*

système est une structure arborescente de fichiers (files) et de répertoires (directories). Chaque utilisateur peut créer à volonté dans son espace de travail de nouveaux fichiers et répertoires.



3.1 Fichiers et répertoires

Notions importantes:

/ est le nom du répertoire racine de l'arbre.

. désigne le répertoire courant.

.. désigne le répertoire parent (juste au dessus).

Répertoire courant : position dans l'arbre à un instant donné.

Répertoire personnel : racine de l'arborescence d'un utilisateur (home directory) et répertoire courant à chaque connexion. Par exemple /home/gromit est le répertoire personnel de gromit.

Nom absolu : désigne de manière unique un fichier en partant de la racine. Chaque fichier a un nom absolu unique dans le système. Par exemple /home/wallace/rep1/f3

Nom relatif : désigne un fichier à partie du répertoire courant. Par exemple rep1/f3 si le répertoire courant est /home/wallace. Deux fichiers de même nom peuvent coexister s'ils sont dans deux répertoires différents car ils ont un nom absolu différent.

3.2 Naviguer dans l'arborescence

cd *nom_repertoire*	permet de changer de répertoire (**c**hange **d**irectory).
cd ..	permet de remonter dans le répertoire du dessus.
cd	sans argument, la commande vous conduit dans votre répertoire personnel.
mkdir *rep1*	crée le répertoire *rep1* (**m**a**k**e **dir**ectory).
rmdir *rep1*	supprime le répertoire *rep1* (**rem**ote **dir**ectory).

Remarques:

- créer un fichier *f1* dans un répertoire x revient à (i) créer le fichier *f1* et (ii) ajouter le nom *f1* dans la liste de noms contenus dans x
- créer un répertoire y dans un répertoire x revient à (i) ajouter le nom y dans la liste de noms contenus dans x et (ii) créer un fichier y et lui associer une liste vide.

3.3 Droits d'accès aux fichiers

Chaque fichier (ou répertoire) possède un ensemble d'attributs définissant les droits d'accès à ce fichier pour tous les utilisateurs du système.

3.3.1 Classes d'utilisateurs

Il existe 3 classes d'utilisateurs pouvant éventuellement accéder à un fichier :

- le propriétaire du fichier (User).
- le groupe dans lequel appartient le propriétaire (Group).
- les autres (Others).

A sa création, un fichier appartient à son auteur. Le propriétaire du fichier peut ensuite distribuer ou restreindre les droits d'accès sur ce fichier comme on le verra plus loin.

3.3.2 Types d'accès

Pour chaque classe d'utilisateurs, il y a 3 types d'accès à un fichier donné :

- en lecture (Read).
- en écriture (Write).
- en exécution (eXecute).

Au niveau répertoire, ces droits signifient :

- Read : droit de lister les fichiers présents dans ce répertoire.
- Write : droit de créer ou de détruire un fichier qui s'y trouve.
- eXecute : droit de traverser ce répertoire.

3.3.3 Contrôle et visualisation des droits d'accès

Pour cela, on utilise la commande **ls** `-l`.

```
home/wallace> ls -l
total 3
-rw-r--r-- 1 wallace usr 12 Dec 18 12:14 f1
-rw-r--r-- 1 wallace usr 297 Dec 18 12:14 f2
drwxr-xr-x 2 wallace usr 512 Dec 18 12:14 rep1
```

Le premier caractère spécifie si le fichier est un répertoire (caractère `d`) ou un fichier (caractère -). Les 9 caractères suivants identifient les droits d'accès (présence du droit si lettre `r,w` ou `x`) dans l'ordre utilisateur (u), groupe (g) et autres (o).

3.3.4 Modification des droits d'accès

Seul le propriétaire d'un fichier peut modifier ses droits d'accès. Pour cela, il utilise la commande **chmod** avec la syntaxe suivante : **chmod** `qui±quoi` *nom_fichier*
avec `qui=u,g,o,a` et `quoi=r,w,x`, − pour enlever des droits et + pour en rajouter.

Exemples :
`home/wallace>` **chmod** `o-r` *f1*
`home/wallace>` **chmod** `g+w` *f2*
`home/wallace>` **chmod** `go-rx` *rep1*

Le résultat de ces commandes est :
`home/wallace>` **ls** `-l`
total 3
```
-rw-rw---- 1 wallace usr 12 Dec 18 12:14 f1
-rw-rw-r-- 1 wallace usr 297 Dec 18 12:14 f2
drwx------ 2 wallace usr 512 Dec 18 12:14 rep1
```

4 Entrées/sorties et pipe

Généralement, les commandes lisent l'entrée standard et/ou écrivent sur la sortie standard. *Par défaut, l'entrée standard est le clavier et la sortie standard est l'écran.* Il est possible de rediriger ces entrée et sortie standards vers des fichiers à l'aide des caractères > et <.

4.1 Redirection de la sortie standard

Pour que le résultat d'une commande soit rangé dans un fichier au lieu d'apparaître à l'écran, il faut utiliser la syntaxe :

nom_commande [`options`] [*arguments*] > *fichier_sortie*

Si le fichier *fichier_sortie* n'existe pas, il est crée et contiendra le résultat de la commande. S'il existait auparavant, son contenu est est écrasé. Si on veut que ce contenu soit préservé et y ajouter le résultat d'une commande, il faut utiliser la redirection >>

4.2 Redirection de l'entrée standard

De même, au lieu de fournir des données en les entrant au clavier, ces données peuvent être lues dans un fichier avec la syntaxe :

nom_commande [`options`] [*arguments*] < *fichier_sortie*

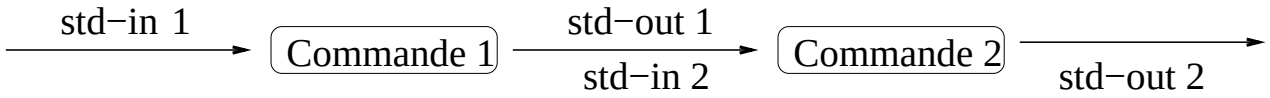
Exemple:
wc `-l` *fich1* : indique à l'écran (sortie standard) le nombre de lignes du fichier fich1.
ls `-l` *fich1* > temp ; **wc** `-l` *temp* : meme résultat que la commande précédente.

4.3 Pipe

Dans le même ordre d'idées que la redirection des entrées/sortie standards, on peut utiliser le *mécanisme de pipe* (tube) qui permet de prendre la sortie standard d'une première commande et de la rediriger sur l'entrée standard d'une seconde commande. La syntaxe utilise le caractère | de la façon suivante :

nom_commande1 [`options`] [*arguments*] | **nom_commande2** [`options`] [*arguments*]

Cet enchainement de commandes peut se shématiser ainsi :



Exemple :
ls `-l` *fich*∗ | **wc** `-l`

5 Outils de recherche

Unix propose de puissant outils de recherche de fichiers ou de recherche dans des fichiers.

5.1 Recherche de chaînes de caractères dans un fichier

La commande **grep** permet d'afficher les lignes des fichiers donnés en arguments qui contiennent un motif donné:

grep `options` motif *fichiers*

Exemple :
grep subroutine **.f*
permet d'afficher à l'écran toutes les lignes des fichiers du répertoire courant ayant comme extension ".f" et contenant le motif "subroutine".

Options élémentaires:
`-i` permet d'ignorer la distinction minuscule/majuscule.
`-v` affichent les lignes qui ne contiennent pas le motif.
`-n` affichent les lignes et leur numéro qui contiennent pas le motif.

5.2 Recherche d'un fichier

La commande **find** descend récursivement dans des sous-arborescences de répertoires, en cherchant à appliquer à des fichiers précisés par un ou plusieurs critères de sélection (nom, type,

date de modification, etc.), une commande donnée. **find** s'utilise de la façon suivante :

find *liste_de_répertoires expression*

où *liste_de_répertoires* est la liste des racines des arborescences à parcourir et *expression* est une suite d'options exprimant les critères de sélection des fichiers et les actions à leur appliquer. Lorsque que le critère est vrai, l'action est exécutée.

Exemple de critères de sélection :

`-name "motif"`	vrai si un nom de fichier courant contien la chaine de caractères motif.
`-user nom_utilisateur`	vrai si un fichier courant appartient à l'utilisateur nom_utilisateur.
`-mtime n`	vrai si le fichier a été modifié dans les n derniers jours (+n pour exprimer "n et plus" et -n pour exprimer "n et moins").

Exemple d'actions à effectuer sur les fichiers sélectionnés :

`-print`	affiche le nom du fichier courant.
`-ls`	liste d'informations sur le fichier courant.

Exemples :
find . -name "*.f" -print
permer de rechercher récursivement à partir du répertoire courant "." tout les fichiers ayant l'extension ".f".

find /home/gromit -name "*" -print | xargs -i grep -i "Wrong Trousers"
permet de recherche récursivement à partir du répertoire /home/gromit tout les fichiers et d'en afficher les lignes qui contienent la chaine de caractères "Wrong Trousers" sans distinguer minuscule/majuscule.