Introduction au système UNIX

Cette première partie décrit quelques commandes de base du système UNIX. Le *shell* est le programme qui s'occupe d'interpréter les "phrases" que vous tapez au clavier en des commandes compréhensibles par la machine. On parle d'interpréteur de commandes. Il en existe généralement plusieurs sur une machine, comme par exemple `/bin/sh` ou encore `/bin/ksh`. Le *prompt* est la phrase qu'affiche l'interpréteur de commandes pour vous dire qu'il est prêt, c'est à dire qu'il attend vos instructions. On considèra dans nos exemples que le prompt est le "$ " —c'est le prompt par défaut du *shell* `/bin/sh`.

On présente ci-dessous quelques commandes de base.

`passwd` Elle permet à un utilisateur de créer ou changer son mot de passe :

```
$ passwd
Changing password for Tome                    (Tome est un utilisateur)
Old password:                                  (Taper l'ancien mot de passe)
Enter new password:                            (Taper le nouveau mot de passe)
Re-enter new password:                        (Confirmer ce nouveau mot de passe)
```

`man` *nom* Cette commande va vous fournir une aide sur la commande de nom *nom*. La première commande que vous pouvez taper pourrait être `man man` ce qui va vous fournir de l'aide sur la commande `man`.

`mkdir` *ref* Crée le répertoire de nom *ref*.

`rmdir` *ref* Supprime le répertoire de nom *ref*.

`rm` *ref* Supprime le fichier de nom *ref*. La commande `rm -i` demande confirmation avant d'effacer les fichiers référencés.

`cd` *ref* Permet de "se déplacer" dans l'arborescence, c'est à dire de passer du répertoire courant au répertoire de nom *ref*. Sans argument le catalogue courant de l'utilisateur devient le catalogue de travail.

`ls` *ref* Sans option, la commande affiche le nom des fichiers du catalogue courant. Avec un ou plusieurs arguments, elle liste les fichiers ou catalogues correspondant à ces arguments.

`cp` *ref1 ref2* Crée une copie physique du fichier de nom *ref1* sous le nom *ref2*.

```
$ ls
fiche
$ cp fiche fiche.old
$ ls
fiche     fiche.old
```

`mv` *ref1 ref2* Change le nom du fichier *ref1* en *ref2*.

```
$ ls -i
256415 fiche       10619 fiche.old
$ mv fiche fichier
$ ls -i
256415 fichier     10619 fiche.old
```

`cat` *ref* Permet d'afficher sur l'écran le contenu du fichier de nom *ref*.

`more` *ref* Affiche, écran par écran, le contenu du fichier de nom *ref*.

Lorsque qu'un programme lit ses instructions au clavier, on dit qu'il les lit sur l'*entrée standard*. De même, lorsqu'il sort des résultats à l'écran, on parle de *sortie standard*. Il est possible de détourner ces entrées et sorties standards via des *redirections*. Cette partie donne un aperçu de tels mécanismes.

com > *ref* Redirige la sortie standard de la commande *com* sur le fichier de nom *ref*. A noter que le fichier est écraser s'il existe déjà.

```
$ ls presents
presents not found
$ who > presents
$ ls presents
presents
```

com >> *ref* Redirige la sortie standard de la commande *com* sur le fichier de nom *ref.* Si le fichier n'existe pas il est créé et sinon, le résulat de l'exécution de la commande est rajouté à la fin du fichier.

```
$ ls presents
presents
$ who >> presents
$ who >> pre
$ ls pre presents
pre       presents
```

com < *ref* Redirige l'entrée standard de la commande *com* sur le fichier *ref*. La commande lira donc ses données dans le fichier au lieu de les lire au clavier.

```
$ who > l1
$ who >> l1
$ wc -l < l1
175
```

com1 | *com2* La notion de `tube` —`pipe` en anglais— permet de rediriger la sortie standard de la commande *com1* sur l'entrée standard de la commande *com2*.

▸ **Exercice 1**. Que pensez-vous de la commande suivante : `who | wc -l`

Éditeurs de texte. . .

Différents éditeurs sont disponibles sur les systèmes. Ils vous permettent par exemple d'éditer des documents, de les modifier mais surtout de les sauver sur le disque. On présente succintement `vi` qui est assez bien adapté à la saisie de programmes —en *C* par exemple. . .

`vi` *ref* Lance `vi` et édite le fichier de nom *ref*.

`vi` dispose d'un mode commande et d'un mode d'insertion. Le mode commande permet de donner des ordres à `vi` concernant par exemple le texte que vous tapez, ou encore le style d'affichage de ce texte; le mode d'insertion vous permet de tapez votre texte.

Les commandes principales au niveau du mode commande :

i	Entrée en mode insertion de texte, avant la position courante.
I	Entrée en mode insertion de texte, avant le premier caractère de la ligne.
a	Entrée en mode insertion de texte, après la position courante.
A	Entrée en mode insertion de texte, en fin de ligne courante.
o	Entrée en mode insertion de texte, après la ligne courante.
O	Entrée en mode insertion de texte, avant la ligne courante.
%	Montre explicitement la correspondance { et } ou encore (et). Cette commande est très utile pour programmer. . .
Esc	Sortie du mode insertion de texte et retour au mode commande.
l	Déplacement du curseur à droite.

h	Déplacement du curseur à gauche.
j	Déplacement du curseur vers le bas.
k	Déplacement du curseur vers le haut.
x	Efface le caractère courant.
dd	Efface la ligne courante.
:se nu	Affiche les numéros de lignes.
:se nonu	Enlève l'affichage des numéros de lignes.
:q	Sortie de `vi`.
:w	Sauvegarde.
:wq	Sauvegarde et sortie de `vi`.
:q!	Force la sortie de `vi` sans sauvegarder.
:x	Sauvegarde et sortie de `vi`.
:e *ref*	Chargement d'un nouveau fichier du nom *ref*.
:f	Donne le numéro de ligne courante.
:*n*	Positionnement à la ligne numéro *n*.

Fichiers de commandes

▸ **Exercice 2**. Un fichier de nom *quote* contient les lignes suivantes :

```
#!/bin/sh
# fichier quote

echo avec rien
echo ls $* *

echo avec \"
echo "ls $* *"

echo avec \'
echo 'ls $* *'

echo avec \`
echo `ls $* *`
```

Que produisent les appels suivants :

```
$ quote
$ quote 1
$ quote -i
```

▸ **Exercice 3**. Ecrire une commande qui affiche chacun de ses paramètres sur une ligne.

▸ **Exercice 4**. Ecrire une commande qui affiche la liste de ses paramètres en ordre inverse.

▸ **Exercice 5**. Ecrire une commande permettant de tester si une variable est un entier.

Présentation du C

Le premier programme que l'on écrit lorsque l'on apprend un nouveau langage est souvent le suivant : "afficher les mots : bonjour maître". En C, on obtient ce résultat par le programme :

```
#include <stdio.h>

void main(void)
{
  printf("bonjour maitre\n");
}
```

Le nom d'un programme en C doit toujours se terminer par l'extension ".c", comme par exemple `bonjour.c`. On le compile en tapant :

```
$ cc bonjour.c
```

Le compilateur va créer un fichier exécutable appelé `a.out` et si on tape la commande `a.out`, ...

```
$ cc bonjour.c
$ a.out
bonjour maitre
$
```

La commande de compilation accepte un certain nombre d'options : par exemple si on avait voulu que l'exécutable ait pour nom `affiche`, on aurait pu taper la commande suivante : `cc -o affiche bonjour.c`.

▸ **Exercice 6**. Comment connaître toutes les options acceptées par la commande `cc`.