

Travaux pratiques d'initiation à Unix

Octobre 2004

Merci à Vincent BOUDET.

1 Manipulation du système de fichiers

Mise en place du TP. Avant toute chose, saisissez la commande

```
% /home/enseign/PDC/bin/init
```

qui met en place un environnement pour votre TP.

Afficher le contenu d'un répertoire. La commande `ls` (list) permet de lister le contenu d'un répertoire. Essayez cette commande en tapant

```
% ls
```

puis

```
% ls -l
```

Consulter l'aide en ligne. Personne n'étant le dépositaire du savoir universel, il est parfois utile de consulter l'aide d'une commande particulière. Pour cela, vous avez à votre disposition la commande `man`. Essayez cette commande en tapant

```
% man man
```

puis

```
% man ls
```

Lire un fichier. Les commandes `more` et `cat` permettent de lire le contenu d'un fichier. Par exemple, tapez la commande :

```
% more .LISEZ-MOI
```

puis la commande

```
% cat .LISEZ-MOI
```

Quelles sont les différences entre ces deux commandes ? Lisez la page de manuel de la commande `more`. Que se passe-t-il si en cours de lecture du contenu d'un fichier avec `more`, on appuie sur la touche `<espace>`, `<entrée>`, `<q>`, `<b>`, ou `</>` ? Il existe aussi une extension de `more` appelée `less` ! Consultez sa page de manuel.

Créer un fichier. Essayez la commande `touch` en tapant

```
% touch foo
```

Utilisez les commandes `ls`, `ls -l` et `more` pour voir l'effet de cette commande. Regardez la page manuel de `touch` pour vérifier votre idée.

Supprimer un fichier. La commande `rm` (remove) permet de détruire un fichier. Essayez cette commande en tapant

```
% rm foo
```

Utilisez `ls` pour voir le contenu du répertoire après destruction.

Attention : la commande `rm` est **DANGEREUSE**. Avant d'utiliser des options magiques comme `rm -fr`, lisez bien la page de manuel, sous peine de détruire tous vos fichiers.

Renommer un fichier. Après avoir lu la page de manuel, testez la commande `mv`. Par exemple, tapez les commandes suivantes :

```
% ls
% touch foo
% ls
% mv foo bar
% ls
```

Copier un fichier. La commande `cp` (copy) permet de faire une copie d'un fichier. Essayez cette commande en copiant le fichier `.LISEZ-MOI` en `toto`. Supprimez le fichier `.LISEZ-MOI`. Vérifiez avec la commande `ls` que le fichier a bien été supprimé. Notez qu'il faut ajouter l'option `a` (a pour all) pour que `ls` liste les fichiers dont le nom commence par un point (`ls -a`). Renommez le fichier `toto` en `.LISEZ-MOI`.

Fichiers et extensions. Souvent, le nom d'un fichier est suivi d'une extension qui, par convention, donne une indication sur le type du fichier. Ainsi, un fichier `foo.c` a de forte chance pour être le source d'un programme C. Des extensions communément utilisées sont par exemple : `.c`, `.txt`, `.log`, `.tex`, `.gz`, `.tar`, `.ps`, etc. Contrairement à ce qui est imposé sur d'autres systèmes de fichiers, sous Unix, les extensions des fichiers sont libres et peuvent même être multiples. On peut nommer un fichier `all.tar.gz`.

Comme vous avez pu le remarquer, les fichiers dont le nom commence par un point ne sont pas listés par la commande `ls` (il faut taper `ls -a`). Ce sont des fichiers cachés que l'on ne souhaite pas voir tout le temps. Typiquement, les fichiers de configuration sont des fichiers cachés. Tapez la commande `ls -a`. Vous devez voir différents fichiers de configuration comme par exemple `.login` ou `.emacs`.

Changer de répertoire courant. La commande `cd` (change directory) permet de changer le répertoire courant. La commande `pwd` affiche ce répertoire courant. Rappelez vous que les fichiers sont organisés dans une arborescence dont le sommet est la racine `/`. Essayez de vous déplacer dans l'arborescence en utilisant des chemins absolus et relatifs. Placez-vous à la racine du système de fichiers. À l'aide des commandes `pwd`, `cd`, et `ls`, parcourez la hiérarchie de fichiers. Identifiez par exemple l'utilité des répertoires `/home`, `/bin`, et `/etc`.

Créer un répertoire. La commande `mkdir` (make directory) permet de créer un répertoire. Le nom du répertoire à créer peut être préfixé par un chemin d'accès. Celui-ci peut être absolu ou relatif.

Supprimer un répertoire. La commande `rmdir` (remove directory) permet de supprimer un répertoire. De même que précédemment, le nom du répertoire peut-être préfixé par le chemin d'accès. Créez plusieurs répertoires et sous répertoires. Utilisez la commande `ls -R` pour visualiser l'arborescence. Supprimez ensuite ces répertoires.

Créer un lien symbolique. La commande `ln` (link) permet de créer un lien entre deux fichiers. Si l'option `-s` est utilisée, le lien créé est dit symbolique. Réfléchissez à l'intérêt d'avoir des liens symboliques. Que se passe-t'il si on efface un fichier référencé par un lien symbolique ?

Complétion. La complétion est un raccourci qu'offre l'interpréteur de commandes (Bash ou Tcsh). La touche `<tab>` est utilisée pour compléter le nom de commandes ou de noms de fichiers. Par exemple, la saisie de

```
% ls .LI
```

suivie de <tab> tente la complétion de `.LI` en un nom de fichier existant, très certainement `.LISEZ-MOI`. Si plusieurs fichiers correspondent au même début de nom saisi, la touche <tab> (Bash) ou <Ctrl-D> (Tcsh) affiche cette liste de fichiers. Essayez avec

```
% ls /bin/l
```

De même, la touche <tab> assure la complétion des noms de commandes. Essayez avec

```
% l
```

2 Noms de fichiers, méta-caractères, et expressions régulières

Deux spécifications des méta-caractères sont utilisées :

1. Les méta-caractères du shell pour identifier des noms de fichiers. Ils sont utilisés directement par le shell et aussi par certaines commandes telle `find`.
2. Les méta-caractères plus généraux utilisés par la plupart des commandes identifiant des expressions régulières.

2.1 Méta-caractères du shell

Méta-caractères du shell. Le shell étend les méta-caractères `?` `*` `[]` `[-]` pour identifier des noms de fichiers qui correspondent à un motif donné. Placez-vous dans le répertoire `/usr/bin`. À l'aide des méta-caractères, listez par la commande `ls` (l'option `-d` de `ls` permet de lister les noms des répertoires plutôt que leur contenu) :

- toutes les entrées dont le nom commence par `t` ;
- toutes les entrées dont le nom comporte exactement trois caractères ;
- toutes les entrées dont le nom comporte au moins trois caractères ;
- toutes les entrées dont le nom comporte au plus trois caractères ;
- toutes les entrées dont le nom comporte au moins un `d` ;
- toutes les entrées dont le nom commence par `a` ou `t` ;
- toutes les entrées dont le nom commence par `r`, `s`, `t` ou `u`.

Recherche de fichiers. La commande `find` permet de retrouver des fichiers. En particulier, il est possible de rechercher des fichiers dont le nom correspond à un motif comportant des méta-caractères du shell. Essayez

```
% find /usr/bin -name '*.sh'
```

2.2 Méta-caractères des expressions régulières

Méta-caractères des expressions régulières. L'ensemble des méta-caractères est le suivant : `\` `|` `( )` `[ { ^ $ * + ?`

- Le caractère `\` transforme un méta-caractère en littéral ;
- le caractère `.` correspond à n'importe quel caractère ;
- le caractère `^` marque le début d'une chaîne ;
- le caractère `$` marque la fin d'une chaîne.

Ainsi

`^bonjour$` reconnaît `bonjour` seul ;

`^$` reconnaît une chaîne vide.

- Le caractère `|` marque l'alternative.

Par exemple,

`L[y|i]s` reconnaît `Lys` ou `Lis` ;

`^(De|Sujet|Date):@` reconnaît tout ce qui commence par `De :@` ou `Sujet :@` ou `Date :@` ;

- Le caractère `*` reconnaît zéro, une ou plusieurs occurrences du caractère précédent ;
- le caractère `+` reconnaît une ou plusieurs occurrences du caractère précédent ;
- l'expression régulière `{x}` avec `x` un nombre reconnaît exactement `x` occurrences du caractère précédent ;
- l'expression régulière `{x,}` reconnaît `x` occurrences et plus du caractère précédent ;
- l'expression régulière `{x, y}` reconnaît au minimum `x` et au maximum `y` occurrences du caractère précédent.

Ainsi,

a? reconnaît 0 ou 1 a ;

a* reconnaît 0, 1 ou plusieurs a ;

a+ reconnaît 1 ou plusieurs a ;

a{2} reconnaît aa ;

a{2, 3} reconnaît aa ou aaa.

- `[ ]` indique une classe où un intervalle dans une classe.

Par exemple :

[a-z] reconnaît les lettres de a à z ;

[Bb]onjour reconnaît Bonjour ou bonjour ;

<h[1-6]> une balise HTML de titre par exemple.

Attention au tiret dans `[ ]`, il représente un intervalle, si on veut l'utiliser comme littéral normal, il faut le mettre après le `[` ou avant le `]`, `[a-z-]`.

- L'expression régulière `[^...]` au lieu de `[...]` indique une classe complémentée.

Par exemple,

[^0-9] reconnaît tout ce qui n'est pas des chiffres ;

[^1-6] reconnaît tout sauf les chiffres de 1 à 6.

Attention, parce que `^` est entre `[ ]`, sa sémantique est différente et ne signifie pas début de ligne !

- L'expression régulière `( )` encadre des instructions pour capturer des sous-chaînes.

Par exemple,

(?ho)+ reconnaît ho, ho ho, ho ho ho, hohoho...

Utilisation des expressions régulières. Certaines commandes, telles `grep` et `sed`, utilisent des expressions régulières pour identifier des motifs. La commande `grep` recherche les lignes de fichiers qui comprennent un motif correspondant à une expression régulière donnée. La syntaxe de `grep` est de la forme

```
grep [options] regexpr [fichiers]
```

Attention, pour éviter une confusion entre les interprétations de ces symboles spéciaux par `grep` ou par le shell, il est indispensable de « verrouiller » `regexpr` en plaçant l'expression entre guillemets `'`.

Voici quelques exemples d'utilisation :

- On cherche dans le fichier `/etc/passwd`, les lignes dont la 1^{re} lettre est quelconque et la 2^e doit être o

```
% grep '^o' /etc/passwd
```

- On cherche dans le fichier `/etc/passwd` les lignes commençant par t

```
% grep "^t" /etc/passwd
```

- On cherche les lignes ne commençant pas commençant par t

```
% grep -v "^t" /etc/passwd
```

Essayez les commandes

```
% grep '-' LISEZ-MOI
% grep '^-' LISEZ-MOI
% grep '^-* ' LISEZ-MOI
```

Notez bien l'espace entre l'étoile et la quote dans cette dernière commande.

Échappement, quotes, double-quotes... Les caractères \ ' " annulent, tout ou en partie, les effets des méta-caractères du shell. Essayez les commandes

```
% echo $USER
% echo "Bonjour $USER"
% echo 'Bonjour $USER'
% echo "* Bonjour $USER *"
% echo '* Bonjour $USER *'
% echo \* Bonjour $USER \*
% echo \* Bonjour \$USER \*
% echo '* Bonjour "$USER"'
```

3 Redirection et entrées/sorties

Lorsqu'une commande a besoin d'afficher quelque chose à l'écran, elle utilise en générale la sortie standard (stdout, numéro 1). Il est possible de rediriger cette sortie standard sur un fichier. Essayez les commandes suivantes :

```
% ls
% ls > foo
% more foo
```

Les erreurs générées par un programme sont elles envoyées sur la sortie erreur (stderr, numéro 2). Il est aussi possible de rediriger cette sortie comme par exemple avec les commandes suivantes (Bourne Shell ou Bash) :

```
% find /etc -name '*'
% find /etc -name '*' 1>foo
% more foo
% find /etc -name '*' 1>foo 2>bar
% more foo
% more bar
```

ou les commandes suivantes (C-shell et sa famille) :

```
% find /etc -name '*'
% find /etc -name '*' > foo
% more foo
% (find /etc -name '*' > foo) >& bar
% more foo
% more bar
```

Vous déterminerez si votre shell est de la famille du Bourne-Shell ou du C-Shell en consultant le contenu de la variable SHELL :

```
% echo $SHELL
```

De même, l'entrée standard (stdin) peut-être redirigée :

```
% wc
la premiere ligne
la seconde
la fin
Ctrl-D
```

```
% wc < .LISEZ-MOI
% wc < .LISEZ-MOI > foo
```

Le *pipe* | permet de rediriger la sortie de l'exécution d'une commande sur l'entrée d'une autre. Par exemple, les commandes :

```
% cut -d: -f1 < /etc/passwd > fa
% sort < fa > fb
```

sont équivalentes à :

```
% cut -d: -f1 < /etc/passwd | sort > fb
```

La commande *diff* permet de lister les différences entre deux fichiers. Utilisez cette commande pour comparer les fichiers *fb* et *fc*.

Notez qu'il est tout à fait possible d'enchaîner plusieurs pipes :

```
% cat /etc/passwd | cut -d: -f1 | sort > fa
```

4 Processus

Lister les processus Chaque processus possède un numéro d'identification unique, son PID. Utilisez les commandes *top* et *ps* pour voir la liste des processus lancés sur votre station.

Tuer un processus. La commande *kill* permet d'envoyer un signal à un processus donné. Après avoir lancé un nouvel *xterm*, essayez de tuer ce le processus associé à cet *xterm* : identification du PID et envoi du signal KILL. Lisez la page de man pour les infos sur la commande *kill*.

Processus séquentiels et concurrents. Tapez la commande suivante :

```
% xterm; emacs
```

Si vous tuez la fenêtre du *xterm*, *emacs* apparaît. Les processus sont séquentiels : dès que l'un a terminé, le suivant commence. Essayez plusieurs combinaisons différentes.

Il est souvent utile de lancer plusieurs processus de manière concurrente. Essayez la commande suivante :

```
% xterm & emacs &
```

Processus en avant plan et arrière plan. La commande *jobs* permet d'avoir la liste des processus lancés à partir du shell courant. Affichez la liste des processus lancés depuis le shell de votre fenêtre.

- Passez *emacs* en avant plan ; voir la commande *fg* ;
- Envoyez le signal *SUSPEND* (*Ctrl-Z*) ;
- Notez que votre *emacs* ne répond plus ;
- Listez les processus avec la commande *jobs* ;
- Remettez *emacs* en avant plan ;
- Envoyez le signal *KILL* (*Ctrl-C*).

5 Droits et sécurité

Permission sur les fichiers. Utilisez la commande *ls -l* pour afficher les permissions associées aux fichiers du répertoire courant. La commande *chmod* permet de modifier les permissions d'un fichier donné. Regardez la page *man* et changez les droits du fichier *.LISEZ-MOI* en lecture seule pour le propriétaire.

Changer d'identité. La commande `su` permet de changer d'identité dans un shell. Prenez l'identité du deuxième binôme puis essayez de lire le fichier `.LISEZ-MOI` précédent.

Changer de mot de passe. Utilisez la commande `passwd` pour changer votre mot de passe.

6 Emacs

Commandes de base. Allez visiter les différents menus d'`emacs`, et essayez d'ouvrir un fichier, de le sauver, de le fermer, de le sauver sous un autre nom, etc. Regardez les raccourcis clavier associés à ces actions et essayez les. Vous pouvez aussi consulter la carte de référence d'`emacs` disponible à partir de www.refcards.com.

Ctrl-x Ctrl-f ouvrir un fichier

Ctrl-x Ctrl-s sauver

Ctrl-x Ctrl-w sauver sous

Ctrl-x Ctrl-c quitter emacs

Ctrl-K supprime tous les caractères jusqu'à la fin de la ligne

Sélection à la souris puis Ctrl-W supprime la zone sélectionnée.

Ctrl-space Pour activer le debut d'une sélection

Ctrl-s recherche en avant

Ctrl-r recherche en arrière

Ctrl-g annuler la dernière commande

Meta-/ complétion automatique

Ctrl-_ undo circulaire

Ctrl-x b passer d'un buffer à un autre

Ctrl-x k tuer un buffer