TP2 : Unix et les processus

1 Et pour quelques commandes de plus ...

La plus importante de toutes les commandes : `man`. N'hésitez pas à l'utiliser.

1.1 Lecture de fichiers

- Créez un fichier test : `ls -l /dev > test`
- Tapez `less test`
- Tapez `head test`. Quelle option utiliser pour afficher non pas les 10 premières lignes, mais les 20 premières ?
- Combien le fichier `test` contient-il de lignes ?
- Faites `ls -l /dev | wc`. Que se passe-t-il ? Conclure sur l'utilité du `|`.
- Quelles différences y a-t-il entre `ls -l /etc | wc` et `ls -l /etc > toto ; wc toto` ?
- Pouvez-vous faire `ls -l /etc | toto` ?
- Faites `ls -l /etc | less` puis `ls -l /etc | less | wc`. Pourquoi ne peut-on pas visualiser le contenu de `/etc` dans le deuxième cas ?
- En utilisant des pipes, `head` et `tail`, afficher uniquement les lignes 10 à 20 de `ls -l /etc`.

1.2 Petites modifications de fichiers

- La commande `tr` est un filtre qui renvoie le flux d'entrée sur la sortie en changeant un caractère en un autre. Faites par exemple :
  `ls -l /dev | tr a q > testb`
  Quel fichier a été créé ? Regardez le début des fichiers `test` et `testb` pour trouver des différences.
- Tapez `ls -l` et notez l'heure de création du fichier `test`. Tapez la commande `touch test`. Quelle est la fonction de cette commande ?
- Tapez `touch testc`. Que s'est-il passé ? Est-ce normal ?

1.3 Commandes utilitaires

- Faites `yes a`. Que se passe-t-il ? Faites Ctrl-C pour arrêter la commande.
- Tapez `echo blabla`. Que fait la commande `echo` ? Que fait `echo blabla > testc` ? Que fait `echo blabla >> testc` ? Que fait `echo blabla | wc` ?

1.4 Chaînage des commandes

Pour exécuter plusieurs commandes à la file en les mettant sur la même ligne, il faut les séparer par des " ; ". On peut aussi parenthéser des commandes ou des séquences de

commandes.

- Tapez : `mkdir my_dir ; cd my_dir ; touch my_file ; ls -l ; rm my_file ; cd .. ; rmdir my_dir`
  Que s'est-il passé ?
- En chaînant `ls` et `tr`, afficher la liste des fichiers de `/etc` en remplaçant les "a" par des "z". Comment peut-on le faire autrement ?
- Tapez `ls ; wc test > resultat` puis `(ls ; wc test) > resultatb`. Comparez les fichiers `resultat` et `resultatb`. Quelle différences a introduit le parenthésage ?

2 Expressions régulières et commandes de recherche

2.1 Aller plus loin avec `ls`

Par défaut `ls rep` affiche tous les fichiers du répertoire `rep`. On peut vouloir n'afficher que certains fichiers et pour cela spécifier ce que l'on veut de la manière suivante :

- "`*`" $\Leftrightarrow$ remplace n'importe quelle suite de caractères (même aucun caractère).
- "`?`" $\Leftrightarrow$ remplace n'importe quel caractère.
- "`[a-f]`" $\Leftrightarrow$ remplace n'importe quel caractère de l'intervalle délimité par `a` et `f`.
- "`[^a-f]`" $\Leftrightarrow$ remplace n'importe quel caractère, sauf ceux de l'intervalle `[a-f]`.

Créez un répertoire `Testls` et allez dedans. Ensuite exécutez la commande :
`yes toto | head -200 | split -5 .`
Que s'est-il passé ? Tout en restant dans ce répertoire, répondre aux questions suivantes :

- Listez le contenu du répertoire.
- Listez tous les fichiers dont le nom finit par "e".
- Listez tous les fichiers dont le nom contient un "a".
- Listez tous les fichiers dont le nom se termine par une lettre comprise entre "c" et "g".
- Listez tous les fichiers dont le nom ne se termine pas par une lettre comprise entre "c" et "g".
- Listez tous les fichiers dont le nom a un "b" en deuxième position.
- Listez tous les fichiers dont le nom a un "b" en deuxième position mais pas de "b" en troisième position.

2.2 Rechercher un fichier

La commande `find` permet de rechercher un fichier dans un répertoire et ses sous-repertoires en basant la recherche sur certains critères (nom, date de création, ...)

- Faire `find /etc -name "fstab"`. A-t-il trouvé un fichier portant le nom `fstab` dans le répertoire `/etc` ? D'où proviennent les erreurs qu'il affiche ?
- Faire `find /usr/include -name "time.h"`. Combien de fichiers correspondant trouve-t-il ?

2.3 Rechercher certaines lignes dans un fichier

La commande `grep` permet de chercher les lignes d'un fichier satisfaisant une certaine propriété, par exemple celles qui contiennent un certain mot, spécifié à l'aide de l'option `-e` :

– Faites `ls -l` puis `ls -l | grep -n -e xb`. Quelles lignes ont été retenues par `grep` ?

`grep` donne à l'utilisateur la possibilité de rechercher non pas des mots précis, mais des mots satisfaisant un motif. Ce motif est spécifié par une expression régulière (*cf.* `man grep` pour une explication plus détaillée). Les expressions ne sont pas tout à fait les mêmes que pour `ls`, faites attention à ne pas confondre.

– "`.`" ⇔ n'importe quel caractère
– "`b*`" ⇔ la lettre "b" répétée autant de fois que nécessaire (y compris 0 fois).
– "`.`" ⇔ n'importe quel caractère.
– "`[a-t]`" ⇔ toute lettre entre "a" et "t".
– "`^`" ⇔ le début de la ligne
– "`$`" ⇔ la fin de la ligne

Il est possible de combiner ces expressions pour faire par exemple `ls -l | grep -e "[a-z]*b$"`, ce qui correspond à rechercher toutes les lignes qui contiennent un mot de longueur quelconque formé de lettres minuscules, puis la lettre "b" juste à la fin de la ligne.

Nota : si l'expression régulière n'est pas précédée d'une option (`-e` par exemple) ni entourée de guillemets, le shell va la traiter comme un ensemble de noms de fichiers. Faites `echo *` et expliquez le résultat.

– Avec `ls -al` et `grep`, afficher une liste de fichiers commençant par "`.`" (*ie* les fichiers cachés, sous unix).
– Refaites les questions de l'exercice 2.1 en utilisant `ls | grep` $\cdots$.

3 Visualisation des processus

Pour voir quels processus tournent sur une machine à un moment donné, il faut utiliser la commande `ps`. La commande `top` affiche de manière répétitive (toutes les secondes par exemple) la liste des processus, avec les statistiques d'utilisation de la mémoire et du processeur ; c'est un `ps` dynamique, en quelque sorte.

3.1 Utilisation de `ps`

– Ouvrir deux terminaux.
– Dans le premier terminal, lancer 2 applications, par exemple acrobat reader et xemacs en faisant :
```
bash> acroread &
bash> xemacs &
```
– Dans le deuxieme terminal, tapez : `bash> ps`
Que se passe-t-il ? Pourquoi `acroread` et `xemacs` n'apparaissent-ils pas dans la liste ? Quelle option utiliser avec `ps` pour les voir ?
– Pourquoi `ps` figure-t-il dans la liste alors que la commande a terminé son exécution ?

3.2 Utilisation de `top`

- Dans le deuxième terminal, lancez `top`.
- Quel processus arrive en tête de la liste ?
- Dans le premier terminal, lancez un `yes` :
  `bash> yes abba`
  Qui arrive en tête maintenant ?

4 Arrêter un processus

C'est le rôle de la commande `kill`.

- Utilisez la commande `ps` pour déterminer le PID (= "Process ID") du `acroread` que vous avez lancé.
- tapez :
  `kill -9 le_pid_d_acroread`
  Que se passe-t-il ?
- Déterminez le PID d'une des commandes `bash` et arrêtez-la avec un `kill -9`. Pourquoi la fenêtre du terminal disparaît-elle ?
- tapez :
  `bash> acroread`
  Pouvez-vous exécuter d'autres commandes dans ce terminal ? Pourquoi ?
- faites un `ctrl-C`. Quel processus a été tué ?

5 Avant-plan et arrière-plan

Pour un terminal donné, seul un processus lancé à partir de ce terminal peut être en avant-plan ; il est le seul à recevoir les commandes tapées au clavier dans ce terminal.

- Lancer un xemacs
  `bash> xemacs`
  Pouvez-vous taper dans la fenêtre d'`xemacs` ? Dans la fenêtre du terminal ?
- Sélectionnez le terminal et faites `ctrl-Z`. Pouvez-vous taper dans `xemacs` ? Dans le terminal ? Pourquoi ?
- Faites `ps -u` . Quel est l'état de `xemacs` ? (colonne `STAT`)
- Faites `bg` puis de nouveau `ps -u`. L'état de `xemacs` a-t-il changé ? Pouvez-vous taper dans `xemacs` ?
- Fermez le `xemacs` ; tapez `xemacs &`. Pouvez-vous tapez dans `xemacs` ? Dans le terminal ? Dans quel état se trouve `xemacs` ? Est-il en avant-plan ou en arrière-plan ?