TP1 en Algorithmique

27 janvier 2009

Un coup d'œil sur shell, rédaction et compilation de programmes

1. Authentifiez vous après du système et lancez une fenêtre shell.
2. Tapez
   ```
   pwd
   ls
   date
   hostname
   ```
3. Tapez `man` suivi par les commandes ci-dessus. `man` (abréviation de "manual") fournit les pages de manuel pour tous les commandes (et les fichier de configuration, et d'autres. On peut faire défiler le texte à l'aide de la barre d'espace, chercher un mot en tapent `/` suivi par les lettres à chercher (`/` tout seul pour chercher la prochaine occurrence), et quitter en tapent `q`. C'est un des commandes le plus importants de shell. Jeter un coup d'œil aux options du commande `ls`, et tapez `ls -l`.
4. `pwd` montre votre position dans la structure arborescent des répertoires. Pour remonter dans cet structure, tapez `cd ..` (`..` est un *alias* pour le répertoire supérieure, `.` pour le répertoire courant). Tapez encore `ls -l`. Tapez `cd` : cela vous remmènera dans votre répertoire d'origine, le répertoire "home" (ce répertoire aussi possède un alias, `~`). Créez un répertoire avec la commande `mkdir` et entrez y en appelant `cd` avec son nom. Essayez a taper exclusivement les premières lettres du répertoire et complétez avec la touche "tab" (⇤⇥). Cette touche "autocomplète" le noms des fichiers et des répertoires. Avec plusieurs possibilités la complétion arrive au plus long préfixe commun. Usuellement (mais pas sur toutes les versions de shell) une nouvelle pression de tab dans ce cas montre les complétions possibles.
5. Assurez-vous d'être à noveau dans votre home. Tapez `ls -l /etc` : l'output sera trop long pour le lire tout. Trois solutions :
   - avec shift+PgUp et PgDn pour naviguer dans l'output de shell ;
   - avec `ls -l /etc > liste.txt`, une introduction aux redirections d'output.
   - avec `ls -l /etc | less`, une introduction aux tubes ;

   redirection : un `> fichier` suivant une commande écrit l'output de la commande sur `fichier`, en effaçant son contenu. `>> fichier` annexe

l'output à la fin du `fichier` sans l'effaçant. On peut regarder le contenu d'un fichier avec la commande `cat` (qui présente le même problème de la longueur), ou `less`, où on peut naviguer comme dans l'output de `man`, ou bien avec un éditeur de votre choix (`kate`, `emacs` ou autres).

tubes : en tapent `commande1 | commande2`, on redirige l'output de `commande1` comme input de `commande2`. Donc `ls -l /etc | less` fait passer l'output du `ls` par `less`.

Faites des épreuves avec plusieurs commandes.

Bonus : essayez

`ls -l /etc | grep conf`,

`ls -l /etc | grep ^d`,

`ls -l /etc | grep conf | less`.

La commande `grep` filtre son input, en en montrant seulement les lignes qui contiennent une *expression régulière* (regexp). "`^d`" est l'expression régulière pour "un d en début de ligne", qui dans `ls -l` est la proprieté des repertoires. Enfin, essayez `ls -lt /etc | head` : cela vous montrera les premières fichiers de `/etc`, triés par leur date de modification (option `-t`).

6. les commandes de gestion des fichiers : `mv` pour déplacer/renommer (fichiers et répertoires), `cp` et `rm` pour copier et effacer respectivement des fichiers, en ajoutant l'option `-r` (recursive) pour les répertoires. Après avoir créé des fichiers terminant en `.txt` et un répertoire `rep`, vous pouvez déplacer tous ces fichiers dans `rep` avec `mv *.txt rep/`. En général `*` est pour une quantité quelconque des caractères (attention, il ne s'agit pas de regexp, mais d'une syntax à part).

   Attention : cela dépende de la configuration de shell, mais usuellement le commandes ci-dessus ne font pas de questions. `mv` et `cp` peuvent souscrire des autres fichiers, et `rm` efface sans passer par une corbeille. Vous pouvez vous habituer à utiliser l'option `-i` (interactive) pour que les commandes demande que faire dans des cas pareils. Si au contraire vous ne voulez aucune question et shell vous en pose, vous pouvez passer l'option `-f` (force). Je repète : on a pas une courbeille, effacer c'est pour toujours.

   Bonus : juste pour avoir un échantillon de la potentialité de shell. On a le répertoire `rep` : essayez la commande `tar czf rep-$(date +%F) rep`. Cette commande archive tout le contenu du répertoire `rep` dans un archive nommé `rep-2009-01-27.tar.gz`, avec la date d'aujourd'hui, parfait pour faire des copies de sauvegarde. On peut le décomprimer avec la commande `tar xzf` suivi par le nom de l'archive. Avec le même system, on peut créer un repertoire avec `mkdir tp-algo-$(date +%m-%d)`...

7. `cat` fournit aussi une façon d'écrire vite un fichier. Essayez à taper `cat > fichier.txt`, et commencez à écrire des lignes. Pour sortir tapez ctrl+d. Contrôlez le contenu du fichier.txt. Une autre manière est avec `echo` : essayez maintenant `echo "epreuve" >> fichier.txt`.

   Bonus : On peut aussi faire de la redirecition de l'input. Essayez cettes deux ligne de commande : `echo "/etc" > fichier.txt`, et après `ls -l < fichier.txt`. Cela est très utile pour faire des épreuves rapides sur des programmes, en écrivant l'input une seule fois dans un fichier.

Un détaillé mais pas trop long tutorial sur l'utilisation de la shell peut être trouvé sur `http://www.grymoire.com/Unix/Sh.html`. Avis : cela va au delà des objectifs du cours.

Premières programmes

1. Temps de saluer le monde. Ouvrez votre éditeur favoris (utilisez `kate` si vous n'en avez un), et créez un fichier `bonjour.c` (dans l'interface de l'éditeur ou en l'appelant avec ce nom, comme `kate bonjour.c`). Ouvrez le programme avec `kate &` ou `kate bonjour.c &` (ou l'éditeur que vous utilisez) : avec la & finale la shell reste active. Écrivez dans `bonjour.c` :

```
1 #include <stdio.h>
2 main() {
3         printf("Bonjour monde !\n");
4 }
```

   Dans la shell, tapez `gcc bonojur.c`. Félicitations, vous avez compilé un programme. Avec `ls` vous pouvez contrôler le nouvel fichier, nommé par défaut `a.out`. Vous pouvez changer ce comportament en tapent `gcc bonjour.c -o bonjour.out`. Ce fichier est la traduction en langage machine de votre code en langage C.

   Pour exécuter votre programme, vous le devez appeler par `./bonjour.out`.

2. Que est-ce qu'il passe si vous enlevez le point et virgule dans la source et essayez à le recompiler ? Et si vous enlevez une "r" de "printf" ? Et si vous enlevez la première ligne ? Et si vous enlevez "\n" ?

3. Créez un autre source, `bonjour-age.c`, avec le code suivant

```
1 #include <stdio.h>
2 main() {
3         int n;
4         printf("Bonjour, quel est votre an de naissance ? ");
5         scanf("%d",&n);
6         printf("Alors vous aurez %d ans a la fin de cette
              annee.\n",2009-n);
7 }
```

   Compilez et testez-le. Que est-ce qu'il passe si vous insérez des lettres à la place d'un entier ?

4. Écrire un programme qui prend en entrée deux entiers et en affiche la somme.

5. Écrire un programme qui prend en entrée deux réels et en affiche la moyenne harmonique (égale à $(x^{-1} + y^{-1})^{-1}$).

6. Utiliser un cycle `for` pour écrire un programme qui prend en entré un entier n, et après demande n entiers et en affiche la somme.

7. Utiliser un cycle `while` pour écrire un programme qui prend en entré des entiers jusqu'à que on n'en insère un négatif. On en affiche la somme, exclu celui négatif.

   (Essayez à écrire un fichier `fichier.txt`, contenant `3 5 7 2 -5`, et exécutez le programme avec `./a.out < fichier.txt` pour voir comme marche la rédirection de l'input dans ce cas)