TP1

La page web du cours se trouve ici :

`http://www.lsv.ens-cachan.fr/~schwoon/enseignement/systemes/ws1516/`.

Vous y trouverez les transparents du cours ainsi que certains fichiers supplémentaires pour les exercices.

1 La ligne de commande (shell)

Les exercices suivantes servent à vous familiariser avec le *shell* de Linux.

1. Lancez une console, puis essayez le suivant :
   — Créez un nouveau répertoire (`mkdir test`) et y aller (`cd test`).
   — Créez un fichier `toto`) avec le texte (`Bonjour`) dans ce répertoire (à partir de l'interface graphique).
   — Dans le shell, vérifiez si le fichier a bien été crée en affichant la liste des fichiers (`ls`) et affichez le contenu du fichier (`cat toto`).
   — Renommenz le fichier (`mv toto tata`).
   — Créez un sous-répertoire de `test` qui s'appele `tmp` et recopiez-y votre fichier (`cp tata tmp/`).
   — Détruire le fichier dans ce sous-répertoire (`rm tmp/tata`) et le sous-répertoire (`rmdir tmp`).
2. Un petit peu plus avancé : création des scripts
   — Créez un fichier (appelé `script`) qui contient quelques commandes à exécuter (p.ex. copier ou afficher des fichiers, afficher un fichier, etc.). La toute première ligne de ce fichier doit être `#!/bin/bash`.
   — Une fois le fichier crée, rendez-le *exécutable* (`chmod +x script`).
   — Vous pouvez désormais exécuter ce fichier en faisant `./script`.
   — Dans `script`, les chaînes `$1`, `$2` etc seront remplacées par le première, deuxième, etc argument donné lors d'une invocation. P.ex., dans `./script toto $1` sera remplacé par `toto`. Servez-vous-en pour créer un script parametré.
3. Quasiment toutes les programmes disponibles dans le shell connaissent un nombre d'arguments modifiant leur comportement. P.ex. `ls -l` donnera plus de détails sur les fichiers. Le programme `man` donne toutes ces informations. Utilisez, p.ex., `man ls` pour trouver comment afficher des fichiers « cachés »qui commencent avec un point (`.`). Il vous est fortement recommandé de consulter ces "pages man" pour toutes les commandes que vous allez utiliser.

2 Quelques informations

Expressions globales Les `expressions globales` peuvent être utilisés sur la ligne de commande pour travailler avec de multiples fichiers à la fois. Le point d'interrogation (`?`) répresente n'importe quel caractère. L'étoile (`*`) répresente une séquence (peut-être vide) de caractères aléatoires. Avant d'exécuter un programme, le shell remplacera une expression globale par la liste des fichiers qui vont avec cette expression (s'il en existe au moins un).

Enchaînement de commandes

— `cmd1 ; cmd2` – exécuter `cmd1` puis `cmd2`.
— `cmd1 | cmd2` – exécuter `cmd1` et passer le texte produit à `cmd2`.

Boucles :

— `for i in liste ; do cmds ; done` : exécuter `cmds` pour toutes les valeurs dans `liste` (valeurs séparés par espaces, p.ex. produites par une expression globale) ; la variable `$i` contient la valeur actulle à l'intérieur de `cmds`.

3 Quelques exercices avancées

1. Télécharger l'archive `tp1files.zip` de la page web du cours et extraire ses contenus.
2. La commande `convert file1 -thumbnail 200x200 file2` créera une image plus petite à partir de `file1`. Écrivez une commande `for` qui le fera pour les images trouvées dans l'archive. E.g., créez `petit-be.jpg` à partir de `be.jpg`.
3. Le fichier `lesbleus.txt` contient les résultat de l'équipe de foot français depuis sa fondation, avec les dates, l'opposant et le statut du match (`A` pour amical, `CM` pour coup du monde, etc), avec les buts français suivi par les buts de l'autre équipe. Trouvez comment utiliser `sort` pour trouver les matchs avec le plus de nombre de buts français. Utiliser `grep` pour n'afficher que les match de la coupe du monde, ou tous sauf les matchs amicaux. Trouver combien de matchs ont été joué (`wc`). Combiner ces analyses.

4 Programming the IAS

L'URL suivant contient un simulateur pour l'IAS, téléchargez et installez-le.

`http://www.cs.colby.edu/djskrien/IASSim/`

Avec les instructions données pendant le cours et sur la page web, équivez quelques programmes :

1. Un programme simple qui lit deux mots dans la mémoire et stocke leur somme dans une troisième position.
2. Un programme qui calcule la somme de tous les mots dans un bloque de mémoire séquentiel, jusqu'à ce qu'il trouve la valeur 0.
3. Calculer les carrées i^2, for $i = 1, \ldots, n$ (où n est un paramètre stocké dans la mémoire) est les stocker dans la mémoire.