DIU Enseigner l'Informatique au Lycée
Bloc 3

Systèmes d'exploitation

Consignes les exercices ou questions marqués d'un ⋆ devront être d'abord rédigés sur papier (afin de se préparer aux épreuves écrites du partiel et de l'examen). En particulier, il est recommendé d'être dans les même conditions qu'en examen : pas de document ni de calculatrice. Tous les TPs se font sous Linux.

1 Commandes de base

⋆ On considère la séquence de commandes ci-après, rentrées les unes après les autres dans un terminal. Décrire l'effet de chaque commande (création de répertoire, changement de répertoire, affichage dans le terminal, erreur, . . .).

1. `cd ~`
2. `mkdir UnixProgWeb`
3. `mkdir UnixProgWeb/TP1`
4. `cd UnixProgWeb/TP1`
5. `cd ..`
6. `ls`
7. `chmod u+rwx,g-rwx,o-rwx TP1`

Ouvrir un terminal et effectuer les commandes ci-dessus. Constatez que tout se déroule comme prévu.

2 Expressions régulières

⋆ pour chacune des expressions régulières ci-dessous, donner une suite de caractères de longueur au moins 1 reconnue par l'expression.

1. `*txt`
2. `+(txt)`
3. `[0-9]*`
4. `+([0-9])`
5. `@(a.txt | b.txt | c.txt)`
6. `+([^0-9])+([0-9])+([^0-9]).bak`
7. `????`
8. `?*?`

Tester les réponses dans le terminal. Pour cela, se placer dans le répertoire `TP1` créé lors de l'exercice précédent, créer des fichiers vides au moyen de la commande « `touch` *nomdefichier* » et tester l'expression au moyen de « `ls` *expression* ». Le fichier que vous venez de créer doit être listé (au moins lui). Exemple :

```
$ touch toto.txt
$ ls *txt
```

Affiche `toto.txt`.

3 Permissions

⋆ On suppose pour cet exercice que le répertoire courant est le répertoire personnel. De plus, on suppose que les répertoires `UnixProgWeb` et `UnixProgWeb/TP1` existent (cf. exercice 1).

Donner la commande permettant de mettre les permissions demandées. On utilisera des permissions numériques et on justifiera en montrant la représentation binaire.

1. le répertoire personnel possède tous les droits pour l'utilisateur et uniquement le droit d'exécution pour le groupe et les autres
2. les répetoire `UnixProgWeb` et `UnixProgWeb/TP1` possèdent tous les droits pour l'utilisateur et les droits de lecture et d'exécution pour le groupe et les autres (une commande par répertoire)

3. le fichier `lisible.txt` du répertoire `UnixProgWeb/TP1` possède les droits de lecture/écriture pour l'utilisateur et uniquement les droits de lectures pour le groupe et les autres
4. le fichier `secret.txt` du répertoire `UnixProgWeb/TP1` possède les droits de lecture/écriture pour l'utilisateur et aucun droit le groupe et les autres

Mettre les permissions sur tous les fichiers et répertoires comme indiqué ci-dessus (vous pouvez créer 2 fichier `lisible.txt` et `secret.txt` au moyen d'un éditeur de texte). Demander à un voisin (sur une autre machine) de tester les permissions de vos répertoires et fichiers (en essayant de rentrer dans les repertoires et de lire les fichiers). En particulier constater la différence entre droit en exécution et droit en lecture sur un répertoire. Enfin, supprimer les droits pour tout le monde et le groupe sur le répertoire personnel. Constater que plus personne n'a accès à ce répertoire à part le propriétaire.

4 Manipulations de fichier textes

Afficher au moyen de la commande `cat` le fichier `/etc/passwd` (ce dernier contient la liste des utilisateurs « locaux » de la machine, *i.e.* pas les utilisateurs dont les comptes sont en réseaux comme les profs ou les étudiants). Pour chacune des actions suivantes, on se référera au cours 1 et éventuellement à la page de manuel de la commande en question si le cours ne donne pas de détail [1]

Remarque il est vivement conséillé d'ouvrir un fichier texte et d'y copier/coller les commandes que vous essayez afin de garder une trace de ce que vous avez fait.

1. afficher les 5 premières lignes du fichier `/etc/passwd`
2. afficher la page de manuel de la commande `tac`
3. utiliser la commande `tac` pour afficher le fichier `/etc/passwd` à l'envers
4. trier le fichier `/etc/passwd`. Quel est l'ordre utilisé ?
5. lire attentivement la page de manuel pour trier selon le troisième champs. On cherchera quelle option permet de spécifier le séparateur de champs et quelle option permet de donner un numéro de champ. Quel est l'ordre utilisé ?
6. trier selon le même champ que la question précédante mais en utilisant l'ordre numérique (chercher dans la page de manuel). En quoi l'ordre est-il modifié ? donner deux lignes dont l'ordre l'une par rapport à l'autre à changé.

Récupérer le fichier se trouvant à l'adresse suivante :

`https://www.lri.fr/~kn/teaching/upw/td01/cdm.txt`

(en utilisant un navigateur Web) et le sauver dans le repértoire `TP1`. Afficher le contenu du fichier. Ce dernier contient des listes de matches de phase finale de coupe du monde, sous la forme :

`PAYS1:PAYS2:GROUPE`

(où `GROUPE` est une lettre entre `A` et `H`). On souhaite déterminer combien d'équipe distinctes sont dans le fichier.

1. lire la page de manuel de la commande `cut`. Déterminer commant extraire uniquement la première « colonne » du fichier `cdm.txt` (*i.e.* déterminer commant afficher uniquement le premier champ de chaque ligne, en spécifiant que le délimiteur de champ est «`:` »
2. Sauver le résultat de la première commande dans un fichier `liste1.txt` on peut sauver la sortie d'une ligne de commande en écrivant `commande arg1 ... argn > fichier`. Sauver de la même manière la deuxième colonne du fichier `cdm.txt` dans un fichier `liste2.txt`
3. sachant que le « | » permet d'enchaîner deux commandes en passant la sortie de la première comme entrée à la seconde, utilisez dans un premier temps la commande `cat` pour afficher les fichiers `liste1.txt` et `liste2.txt` l'un à la suite de l'autre puis enchainer le résultat avec la commande `sort`.
4. consulter (encore) la page de manuel de `sort` pour trouver l'option qui permet de supprimer les doublons. Afficher la liste des pays sans doublons
5. consulter la page de manuel de la commande `wc` (*word count*) pour savoir comment compter le nombre de lignes d'un fichier. En déduire rajouter la bonne invocation de la commande `wc` à votre enchainement pour déterminer le nombre de pays.

1. si la page de manuel n'est pas disponible sur votre machine, vous pouvez consulter l'url suivante : `http://www.linux-france.org/article/man-fr/man1/Index-1.html`

5 Recherche de fichiers

On suppose que le répertoire courant est le répertoire utilisateur et que les fichiers des exercices précédents n'ont pas été effacés (`liste1.txt`, `liste2.txt`, `secret.txt`, ...).

1. Utiliser la commande `find` pour trouver tous les fichiers dont l'extension est `.txt` et se trouvant quelque part dans le répertoire utilisateur (ou un sous-répertoire).
2. Lire la page de manuel de la commande `xargs`. Cette dernière prend la forme `xargs commande` et lit en plus *n* noms de fichiers sur son entrée standard qu'elle passe comme arguments à commande. Enchainer (au moyen de « | ») les commandes `find` et `xargs cat` pour afficher dans le terminal toutes les lignes de tous le fichiers texte.
3. Comment compter le nombre de lignes ainsi affichées ?

6 Redirections

Exercice 1

⋆ Pour chacune des questions suivantes, donner les commandes permettant de réaliser les actions demandées. On cherchera d'abord les commandes en s'aidant du cours ou des pages de manuel.

1. Se placer dans le répertoire `UnixProgWeb` créé au TP1. Créer un sous-répertoire TP2 et se placer dedans.
2. Lister de façon détaillée le contenu du répertoire. Dire maintenant comment créer un fichier `liste.txt` contenant cette liste détaillée.
3. Lister de façon détaillée les fichiers `liste.txt` ainsi qu'un fichier `pasla.txt` (inexistant). Quel est l'affichage de la commande ?
4. Effectuer la même commande en redirigeant les erreurs vers un fichier `erreur.txt` et la sortie standard vers un fichier `liste2.txt`. Afficher tour à tour (en 2 commandes) le contenu de ces fichiers.

Exercice 2

La commande `dc` implémente une calculatrice en notation *polonaise inversée* (les opérandes sont placées sur une pile et les opérations dépilent les opérandes et empilent le résultat). Elle attend des ordres de calcul sur son entrée standard et les exécute. Par exemple (les lignes en *italique* sont celles à taper au clavier, les lignes en **gras** sont celles affichées par le programme) :

```
$ dc
42                ← invocation de la commande
41                ← place l'entier 42 sur la pile
+                 ← place l'entier 41 sur la pile
p                 ← dépile deux nombres et place leur somme sur la pile
83                ← affiche le sommet de pile
```

1. au moyen d'un éditeur de texte, créer un fichier `calculs.txt` contenant les lignes :

```
40
260
+
10
/
p
12
*
p
```

2. ⋆ Que serait l'affichage produit si cette série de commandes étaient données à `dc`. Donner *deux* lignes de commandes différentes permettant de passer ce fichier en entrée à `dc`.

7 Gestion des processus

Sous Unix, les processus peuvent avoir l'état suivant :

```
R      en cours d'exécution
D      en attente, non interruptible (généralement en train de faire une entrée/sortie)
S      en attente, interruptible
T      arrêté
Z    « zombie », processus défectueux, terminé mais dont les ressources ne sont pas encore libérées
```

Il peut de plus y avoir des informations complémentaires :

```
+      le processus est en avant plan
s      le processus est un leader de session (usuellement c'est le shell)
l      le processus est en fait un thread (sous-processus)
```

⋆ Fermez toutes vos fenêtres et ne conservez qu'une seule fenêtre de terminal. Tapez la commande `ps u`. Quels sont les programmes en cours d'exécution et quels sont leurs états.

8 Scripts *bash*

Exercice 4

On considère le script suivant :

```
#!/bin/bash
for i in $(seq 1 10)
do
  if test -d "TP$i"
  then
     echo "TP$i" "existe déjà"
  else
     echo "création de TP$i"
     mkdir "TP$i"
  fi
done
```

1. Créez un fichier (par exemple `script.sh`) dans le répertoire `UnixProgWeb`. Placez dans ce fichier le contenu du script ci-dessus (la première ligne du script doit être celle commençant par `#!...` sans ligne vide ou espace avant. Essayez d'exécuter le script en effectuant la commande : `./script.sh` (le répertoire courant dans votre terminal doit être `UnixProgWeb`). Que se passe-t-il ? Comment régler le problème ? Réglez le problème et exécutez la commande.
2. Commentez ce script (les commentaires en shell commencent par un # et finissent à la fin de la ligne), en décrivant **précisément** ce qui se passe à chaque ligne.

Exercice 5

On souhaite écrire un script bash qui automatise le téléchargement de fichiers à une URL connue.

1. Placez vous dans le répertoire `UnixProgWeb/TP2`. Récupérez **sans utiliser de navigateur web** le fichier se trouvant à l'url « `https://www.lri.fr/~kn/php/cm/pays.txt` ». Pour ce faire, utilisez la commande `wget` de cette manière :

   ```
   $ wget -nd -nc -q "https://www.lri.fr/~kn/php/cm/pays.txt"
   ```

   L'option `-nd` récupère le fichier et ne crée pas les répertoires `kn/php/...` dans le répertoire courant. L'option `-nc` évite de retélécharger le fichier s'il est déjà dans le répertoire courant. L'option `-q` supprime les messages de débuggage de la commande. Constatez que le fichier a bien été rappatrié dans le répertoire courant et affichez son contenu avec `cat`.
2. Donner une commande permettant de récupérer le troisième champ (séparé par des « | ») du fichier `pays.txt` (attention le caractère « | » est un caractère spécial, il convient de l'échapper convenablement sur la ligne de commande).
3. Écrire un script shell qui télécharges tous les fichiers d'images listés dans le fichier `pays.txt` (le troisième champ donné par la question précédante), sachant qu'ils se trouvent à l'adresse : « `https://www.lri.fr/~kn/php/cm/`*fichier.png* ». Lancez votre script et vérifier que les images sont bien rappatriées. Si le fichier existe déjà, votre script doit le supprimer avant de le retélécharger.
4. (facultatif) Modifier le script précédant pour qu'il affiche en fin d'exécution le nombre de fichier téléchargés ainsi que la somme des tailles des fichiers téléchargés (en octets).