Réseaux — Partie 3 : Exercices

IUT de Villetaneuse — R&T 1ère année

Laure Petrucci

19 janvier 2011

1 Environnement Unix

Exercice 1.1 : Commandes de base

Question 1 : Créer un répertoire `TPR3` et s'y placer. Ce sera votre répertoire de travail pour tous le module.

Question 2 : Créer, en utilisant la commande `cat`, un fichier `essai` contenant le texte suivant :

```
Ceci est un essai.
```

Question 3 : Afficher le contenu du fichier `essai` avec une commande autre que `cat`.

Question 4 : Créer un répertoire `fichvides` contenant deux fichiers vides `vide1` et `vide2`. Pour créer ces fichiers, on utilisera la commande `touch` qui met à jour les dates de modification et d'accès à un fichier, et le crée si celui-ci n'existe pas.

Question 5 : Peut-on supprimer le répertoire `fichvides` avec la commande `rmdir` ? Pourquoi ?

Question 6 : Supprimer tous les fichiers dans le répertoire `fichvides` mais pas ce répertoire.

Question 7 : Peut-on supprimer le répertoire `fichvides` avec la commande `rmdir` ? Pourquoi ?

Exercice 1.2 : Types de commandes

Question 1 : Créer un alias `cherche` qui recherche dans l'arborescence privée de l'utilisateur tous les fichiers de nom `essai`.

Question 2 : Que fait `which` ? Quel est son type ?

Question 3 : Que fait `kill` ? Quel est son type ?

Question 4 : Quel est le type de `cherche` ?

Question 5 : Quelle est la liste des alias définis ?

Question 6 : Où se trouve la commande `rm` ? Quel est on type ? Que peut-on en déduire ?

Exercice 1.3 : Utilisation de l'historique

Question 1 : Affichez l'historique des commandes que vous avez utilisées.

Question 2 : Quelle est la taille de l'historique ?

Question 3 : Quel est le numéro de la dernière commande `cherche` utilisée ?

Question 4 : Répéter l'exécution de cette commande — à partir de l'historique — en utilisant son numéro.

Question 5 : Répéter l'exécution de cette commande — à partir de l'historique — en utilisant le début de son nom.

Exercice 1.4 : Pile de répertoires

On souhaite créer une commande permettant de revenir à un répertoire dans lequel on se trouvait précédemment. Pour cela, on utilisera les commandes internes `dirs`, `pushd` et `popd`.

Question 1 : La commande `dirs` affiche une pile de répertoires. Une pile est une structure qui a un comportement similaire à une pile de livres (pour prendre un exemple pratique). Quand on ajoute un élément, on dit qu'on l'*empile* (*push*), et il se retrouve au *sommet* de la pile. Lorsqu'on retire un élément, celui du sommet est enlevé, et on dit qu'on *dépile* (*pop*).

Expliquer les effets de la suite de commandes suivante sur la pile de répertoires :

```
pwd
dirs
pushd ..
pwd
dirs
popd
pwd
dirs
```

Question 2 : Écrire un alias pour que la commande `cd` « se souvienne » du répertoire que l'on quitte.

Question 3 : Écrire une commande `back` qui retourne dans le répertoire précédent.

2 Éditeurs standards

Exercice 2.1 : Utilisation de `vi`

Question 1 : Enregistrer la liste détaillée des fichiers de `/usr/include` dans un fichier `include.txt`. puis éditer ce fichier en utilisant la commande `vi`.

Question 2 : Effacer la première ligne de ce fichier.

Question 3 : Répéter cette opération.

Question 4 : Annuler cette suppression de ligne.

Question 5 : Ajouter en début de fichier :

```
Fichiers de /usr/include :
```

et sauvegarder le fichier modifié.

Question 6 : Aller — en une seule commande — en fin de fichier, et ajouter :

```
Fin du fichier
```

Question 7 : Quitter l'éditeur sans enregistrer les dernières modifications.

Exercice 2.2 : Utilisation de commande de l'éditeur `ed` dans `vi`

Éditer à nouveau le fichier `include.txt` et supprimer toutes les lignes correspondant à un répertoire, en une seule commande. Puis sauvegarder le résultat.

3 Variables d'environnement

Exercice 3.1 : Modification de variables d'environnement

Question 1 : Modifier la variable `PATH` pour que les fichiers de votre répertoire de travail puissent être exécutés en priorité.

Question 2 : Modifier votre message d'invite pour qu'il soit de la forme :

```
utilisateur@machine:répertoire à heure >
```

Exercice 3.2 : Fichier de configuration

Utiliser l'éditeur de texte `vi` pour personnaliser le fichier de configuration de façon que les exécutables soient toujours recherchés en priorité dans le répertoire de travail.

4 Scripts *shell*

Exercice 4.1 : Reformatage de la date

Question 1 : Faites afficher la date.

Question 2 : Écrivez un script *shell* qui affiche sur une première ligne dans l'ordre les jour, quantième, mois, année, puis sur une deuxième ligne l'heure.

Exercice 4.2 : Script *shell* : suppression récursive

Question 1 : Écrivez une commande prenant comme arguments des noms de fichiers, ayant le même effet que *rm -i*. On n'effacera pas réellement les fichiers, mais on affichera un message correspondant à l'opération à effectuer dans chaque cas.

Question 2 : Modifiez le script de la question 1 pour prendre en compte les cas où les arguments sont des répertoires ou des fichiers inexistants.

Question 3 : Modifier le script de la question 2 pour détruire récursivement les répertoires.

5 Gestion des utilisateurs

Exercice 5.1 : Création d'un nouvel utilisateur

Question 1 : Créer, le plus simplement possible, un utilisateur `R3user1` de numéro 1234 ayant pour *shell* `/bin/bash`.

Question 2 : Consulter le contenu des fichiers `/etc/passwd` et `/etc/shadow`. Que remarque-t-on ? Vérifier que le répertoire privé de l'utilisateur a été créé.

Question 3 : Affecter le mot de passe `passR3` à cet utilisateur.

Question 4 : Consulter le contenu des fichiers `/etc/passwd` et `/etc/shadow`. Que remarque-t-on ?

Exercice 5.2 : Modification de groupes

Question 1 : Consulter le fichier `/etc/group`. Que remarque-t-on ?

Question 2 : Créer un groupe `R3`.

Question 3 : Consulter le fichier `/etc/group`. Que remarque-t-on ?

Question 4 : Supprimer le groupe de l'utilisateur `R3user1`.

Question 5 : Modifier le groupe principal de l'utilisateur `R3user1` en `R3`, à l'aide de la commande `usermod`.

Question 6 : Consulter les fichiers `/etc/passwd` et `/etc/group`. Que remarque-t-on ?

Question 7 : Supprimer le groupe initial de l'utilisateur `R3user1`.

Question 8 : Vérifier que le fichier `/etc/group` a bien été modifié.

Exercice 5.3 : Groupes et droits d'accès

Question 1 : Créer un nouvel utilisateur `R3user2` dans le groupe `R3`.

Question 2 : Créer trois groupes : `R312` contenant les utilisateurs `R3user1` et `R3user2`, `R313` contenant l'utilisateur `R3user1`, et `R323` ne contenant pas d'utilisateur.

Question 3 : À l'aide de la commande `usermod`, ajouter l'utilisateur `R3user2` au groupe `R323` tout en gardant `R3` comme groupe principal et `R312` comme groupe secondaire.

Question 4 : Vérifier que le fichier `/etc/group` a bien été modifié.

Question 5 : Créer un utilisateur `R3user3` ayant pour groupe principal `R3` et pour groupes secondaires `R313` et `R323`.

Question 6 : Vérifier que les fichiers `/etc/passwd` et `/etc/group` ont bien été modifiés.

Question 7 : Ouvrir 3 terminaux. En utilisant la commande `su`, se connecter dans l'un des terminaux comme utilisateur `R3user1`, dans un second comme `R3user2`, et dans le dernier comme `R3user3`. Que se passe-t-il et comment résoudre le problème ?

Dans les questions suivantes, on fera référence au terminal dans lequel travailler par le nom de l'utilisateur par lequel l'opération doit être effectuée.

Question 8 : L'utilisateur `R3user1` crée un fichier `testR3` contenant la chaîne de caractères « Test du groupe R3 », et lui attribue les droits d'accès `rw-r-----`.

Question 9 : Les utilisateurs `R3user2` et `R3user3` visualisent le fichier `testR3`. Que se passe-t-il ?

Question 10 : L'utilisateur `R3user1` crée un fichier `testR312` contenant la chaîne de caractères « Test du groupe R312 », l'associe au groupe `R312`, et lui attribue les droits d'accès `rw-r-----`.

Question 11 : Les utilisateurs `R3user2` et `R3user3` visualisent le fichier `testR312`. Que se passe-t-il ?

Exercice 5.4 : Suppression d'utilisateurs et de groupes

Supprimer les utilisateurs `R3user1`, `R3user2` et `R3user3` ainsi que les groupes `R3`, `R312`, `R313` et `R323`, après avoir réfléchi et expliqué l'ordre des opérations.

6 Gestion des processus

Exercice 6.1 : Exécution d'un processus et déconnexion

Utiliser `vi` pour entrer le script *shell* `compter.sh` suivant :

```
#!/bin/bash
i=0
while [ $i -lt 100 ]
do
  echo "i=$i"
  i=$(($i+1))
  sleep 2
done
```

Question 1 : Dans un terminal, exécuter ce script en tâche de fond.

Question 2 : Quitter le terminal. Que se passe-t-il ?

Question 3 : Ouvrir un nouveau terminal et exécuter le script de manière non interruptible lorsque l'on quitte le terminal. Où s'affiche le résultat ?

Exercice 6.2 : Utilisation des signaux

Question 1 : Exécuter le script `compter.sh` dans un terminal. L'interrompre en lui envoyant un signal d'interruption `SIGINT`.

Question 2 : Inhiber le signal `SIGINT` puis relancer `compter.sh` et l'interrompre avec `<CTRL-c>`.

Question 3 : Utiliser `CTRL-z` pour suspendre le processus. Que se passe-t-il ?

Question 4 : Remettre en place la procédure par défaut associée au signal `SIGINT`. Exécuter une nouvelle fois le script `compter.sh` puis l'interrompre avec `<CTRL-c>`. Que se pasee-t-il ?

Question 5 : Relancer en avant-plan le processus suspendu. L'interrompre avec `<CTRL-c>`. Que se passe-t-il ?

Exercice 6.3 : Exécution différée de commandes

Question 1 : Écrire un script *shell* `heure.sh` affichant l'heure.

Question 2 : En utilisant la commande `at`, programmer l'exécution de ce script dans deux minutes. Le résultat devra être écrit dans un fichier `/tmp/heure`.

Question 3 : Recommencer en programmant l'exécution pour 3 heures du matin.

Question 4 : Afficher la liste des tâches programmées et supprimer la dernière.

Exercice 6.4 : Programmation de l'exécution de commandes

Question 1 : Utiliser `crontab` pour programmer l'exécution du script `heure.sh` toutes les minutes pendant l'heure en cours.

Question 2 : Vérifier le bon fonctionnement de la commande.

Question 3 : Supprimer la tâche.

7 Configuration réseau

Exercice 7.1 : Configuration des interfaces réseau

Question 1 : Former un réseau local en reliant deux machines d'une même rangée à un *hub*.

Question 2 : Quelles sont les *adresses MAC* des cartes réseau de la machine ?

Question 3 : Quels sont les constructeurs de ces cartes ?

Question 4 : Proposer une adresse IP pour le réseau local ainsi que des adresses pour les deux machines. Ces dernières adresses prendront la forme `10.10.R.M`, où `R` est le numéro de votre rangée et `M` un numéro de machine dans la rangée.

Question 5 : Quel est le masque de sous-réseau correspondant ?

Question 6 : Utiliser la commande `ifconfig` pour configurer les cartes réseaux avec ces adresses IP.

Exercice 7.2 : Vérification du réseau

On continue à travailler avec le réseau local et les adresses mises en place dans l'exercice précédent.

Question 1 : Lancer la *capture de trames* dans l'outil `wireshark` sur chacune des deux machines.

Question 2 : Utiliser la commande `ping` sur chaque machine pour vérifier qu'elle peut bien atteindre l'autre machine.

Question 3 : Combien de trames ethernet ont été échangées lors du premier envoi de paquet ?

Question 4 : Combien de trames ethernet ont été échangées lors du second envoi de paquet ?

Question 5 : En déduire le fonctionnement du protocole *arp* de résolution d'adresses.

Question 6 : Consulter, à l'aide de la commande `arp` les tables de correspondances entre adresses MAC et adresses IP, puis recommencer quelques minutes plus tard. Que s'est-il passé ?

Question 7 : Attribuer des noms symboliques aux deux machines du réseau local avec `hostname`.

Question 8 : Tester à nouveau le réseau local en fournissant à la commande `ping` l'adresse symbolique au lieu de l'adresse IP des machines.