Systèmes 1
Sujets des Travaux Pratiques

Table des matières

SÉANCE 1

Commandes utiles pour cette séance : `cat`, `cd`, `chmod`, `cp`, `echo`, `grep`, `head`, `ls`, `mkdir`, `more`, `mv`, `pwd`, `rm`, `rmdir`, `sort`, `tail`.

Exercice 1

1). Télécharger et extraire le fichier `users.tar` dans votre répertoire d'accueil. Déplacez vous dans votre répertoire d'accueil. En utilisant la commande `ls`, afficher la liste des fichiers du répertoire `users/linfg/linfg0/S3` dont le nom se termine par `.txt`

2). Afficher intégralement les trois chaînes de caractères suivantes, à l'aide de trois appels à la commande `echo` :

```
Le renard dit "Oh, Monsieur du Corbeau"
J'ai gagne $100 a l'Alcazar
Commentaire : \* Initialisation *\
```

Remarques :

- Le métacaractère \ annihile l'évaluation du caractère immédiatement suivant, ce qui n'est intéressant que dans le cas où ce caractère est un métacaractère du shell.
- Si un délimiteur `'`, `"` ou `` ` `` est « célibataire » sur sa ligne de commande, alors l'interpréteur de commandes affiche un prompt `>` qui invite l'utilisateur à terminer la commande en cours de frappe. La commande n'est réellement interprétée que lorsque les délimiteurs forment des paires.
- En cas de problème, l'exécution d'une commande peut être interrompue en tapant la combinaison de touches <*Ctrl*>C

Exercice 2

Taper les commandes nécessaires à la réalisation de chacune des tâches suivantes :

1). Se déplacer dans votre répertoire d'accueil et afficher le nom du répertoire courant.

2). Créer le répertoire `REP` et son sous-répertoire `SOUSREP`, en une seule ligne de commande.

3). Copier le fichier `users/linfg/linfg0/S3/liste.txt` du répertoire d'accueil dans `SOUSREP`

4). Se positionner dans le répertoire `SOUSREP`, et en afficher le contenu.

5). Déplacer le fichier `liste.txt` de `SOUSREP` vers `REP`

6). Supprimer le répertoire `REP`, ainsi que son contenu.

Exercice 3

Effectuer les tâches suivantes :

1). Se positionner dans le répertoire d'accueil.

2). Copier le fichier `users/linfg/linfg0/S3/liste.txt` dans le répertoire d'accueil, sous le même nom, sans se déplacer.

3). Consulter le contenu de cette copie, sans utiliser d'éditeur de texte (utiliser la commande `more`).

4). Contrôler les droits d'accès au fichier `liste.txt`, en tapant `ls -l`

5). S'en interdire l'accès en écriture.

6). Contrôler la modification des droits d'accès effectuée.

7). Essayer de changer un prénom dans le fichier `liste.txt`, à l'aide de l'éditeur de texte `nedit` ou `gedit` ou `kate`. Sauvegarder. Quitter `nedit` ou `gedit` ou `kate`.

8). Rétablir l'accès en écriture au fichier `liste.txt` pour le propriétaire.

9). Essayer à nouveau d'effectuer la modification précédente, à l'aide de `nedit` ou `gedit` ou `kate` . Sauvegarder.

Exercice 4

La commande `head` permet de récupérer les premières lignes d'un fichier. La commande `tail` permet de récupérer les lignes situées à la fin d'un fichier. Après avoir lancé la commande `man` pour chacune de ces commandes, essayer de trouver *a priori* ce que font les quatre commandes suivantes, puis les taper pour vérifier :

```
tail -n -5 liste.txt
tail -n +5 liste.txt
head -5 liste.txt
head liste.txt
```

Exercice 5

Les lignes du fichier `liste.txt` sont structurées sous la forme suivante (deux champs successifs sont séparés par un caractère espace) :

```
EMMANUEL-EMILE CYRIL linfg151
EMRITTE MEHDI linfg22
```

1). Sachant que tout étudiant a un login du type `linfg<numéro>`, trier le fichier `liste.txt` de manière à afficher la liste des étudiants par ordre de numéros croissants. Pour éviter que le numéro `2` n'apparaisse après le numéro `19`, on conseille de lancer plusieurs commandes successivement, en utilisant l'algorithme suivant :
   - Trier les étudiants dont le numéro ne comporte qu'un chiffre et écrire le résultat dans un fichier `temp`
   - Trier les étudiants dont le numéro comporte deux chiffres exactement, et les rajouter au fichier précédent.
   - Trier les étudiants dont le numéro comporte trois chiffres exactement, et les rajouter au fichier précédent.
   - Afficher le contenu de `temp`, puis détruire ce fichier.

2). Trier ce fichier de manière à afficher la liste des étudiants par ordre alphabétique des prénoms.

Exercice 6

1). Afficher la liste des étudiants ayant `LAURENT` comme prénom.

2). Afficher la liste des étudiants ayant un numéro inférieur à `100` (on aura intérêt à utiliser la commande `grep` avec l'option `-v`).

3). Afficher la liste des étudiants ayant un numéro contenant un chiffre `1` au moins.

Exercice 7

1). À l'aide uniquement de la commande `echo`, et sans utilisation du caractère `\n`, créer le fichier `tdm01.txt` contenant les lignes suivantes :

```
Est-ce que j'arrive
a rediriger la
sortie standard ?
```

2). Déterminer *a priori* quel doit être le résultat de chacune des quatre commandes ci-dessous, puis contrôler :
   - `cat liste.txt`
   - `cat liste.txt > liste2.txt`
   - `rm liste2.txt`

- `cat liste.txt | tail -5`

3). Écrire une commande permettant d'afficher les lignes `5` à `10` du fichier `liste.txt`

SÉANCE 2

Commandes utiles pour cette séance : `cp`, `cut`, `echo`, `grep`, `head`, `ls`, `mkdir`, `rm`, `rmdir`, `sort`, `tail`, `tr`, `wc`.

Exercice 8

Lancer un éditeur de texte, par exemple `nedit` (en tapant la commande `nedit &`). Écrire les scripts permettant de réaliser les trois séquences de tâches suivantes, puis tester ces scripts :

1). `script1.sh`
- Créer, dans le répertoire courant, le répertoire dont le nom est passé en paramètre.
- Recopier un fichier du répertoire courant dans le répertoire créé. Le nom du fichier sera passé en paramètre également.

2). `script2.sh`
- Effacer les fichiers contenus dans un répertoire passé en paramètre.
- Supprimer ce répertoire.

On appellera ce script sur des répertoires créés à l'aide de `script1.sh`

3). `script3.sh`
- Créer un répertoire temporaire `REP`
- Créer dans ce répertoire le fichier `liste.txt`, contenant la liste des noms des fichiers du répertoire courant dont le nom se termine par l'extension `.txt` ou `.sh`
- Recopier le fichier correspondant au dernier nom de cette liste dans `REP`
- Effacer le fichier `liste.txt`

Vérifier que ce script fonctionne bien, en contrôlant l'existence du répertoire `REP` et d'un fichier dans ce répertoire, dont le nom comporte l'extension `.txt` ou `.sh`

Exercice 9

Écrire un script permettant de dénombrer les fichiers du répertoire courant (les sous-répertoires du répertoire courant ne doivent donc pas être comptés) dans les quatre catégories suivantes :
- Fichiers (cachés ou non) dont le nom se termine par `.txt`
- Fichiers (cachés ou non) dont le nom se termine par `.sh`
- Fichiers cachés.
- Autres fichiers.

Chaque résultat devra être affiché sous la forme suivante :
`Il y a 2 fichiers dont le nom se termine par .txt`

Exercice 10

On souhaite effectuer quelques manipulations sur le fichier `liste.txt` qui se trouve dans le répertoire `users/linfg/linfg0/S3/`. Ce fichier avait déjà été utilisé lors de la première séance. Faire une copie de ce fichier dans votre répertoire d'accueil. et répondre aux questions suivantes :

1). Afficher le nom (et uniquement le nom) de l'étudiant de `liste.txt` qui a le plus petit numéro.

2). Afficher le nom (et uniquement le nom) de l'étudiant de `liste.txt` qui a le plus grand numéro.

3). Écrire un script permettant d'afficher le nombre d'étudiants de `liste.txt` dont le nom comporte un nombre de lettres supérieur ou égal à la première valeur et strictement inférieur à la deuxième valeur passées en paramètres.

SÉANCE 3

Commandes utiles pour cette séance : `cat`, `cd`, `cut`, `date`, `echo`, `exit`, `expr`, `grep`, `ls`, `test`, `date`, `pwd`, `read`, `rm`, `set`, `test`.

Exercice 11

Écrire un script permettant de convertir en français la date affichée par la commande « `date -R` », sous le format suivant :
`Lundi 16 Decembre 1996 15:31:37`
Pour l'écriture de ce script, il est conseillé d'utiliser un « aiguillage à choix multiple ».

Exercice 12

Écrire un script qui effectue la traduction mot à mot d'un texte anglais en français. Le fichier `f_anglais.txt` contient le texte écrit en anglais. Le fichier `f_français.txt` contiendra la traduction. On dispose d'un fichier `dico.txt`, dont chaque ligne contient un mot en anglais, suivi d'un espace, suivi de la traduction de ce mot en français. On suppose que `dico.txt` permet la traduction de l'ensemble du vocabulaire utilisé dans `f_anglais.txt`
Exemple :
Si le fichier `f_anglais.txt` contient la ligne :
`my tailor is rich`
et le fichier `dico.txt` contient les lignes :
```
is est
my mon
rich riche
tailor tailleur
```
alors la ligne suivante doit être écrite dans le fichier `f_français.txt` :
`mon tailleur est riche`
Remarque :
Un exemple de fichier `f_anglais.txt` et un fichier `dico.txt` suffisant pour le traduire peuvent être récupérés en tapant :
```
cp users/linfg/linfg0/S3/f_anglais.txt .
cp users/linfg/linfg0/S3/dico.txt .
```

Exercice 13

Écrire un script interactif, permettant de naviguer dans une arborescence de fichiers et de répertoires, selon les spécifications suivantes :

- Le script affiche le répertoire courant et la liste numérotée des sous-répertoires. Les fichiers du répertoire courant sont ignorés.
- L'utilisateur tape le numéro du sous-répertoire dans lequel il souhaite se déplacer, avec les cas particuliers suivants :
  - S'il tape `0` : il reste dans le répertoire courant.
  - S'il tape `-1` : il monte d'un répertoire.
  - S'il tape `-2` : il termine l'exécution du script.

SÉANCE 4

Commandes utiles pour cette séance : `cat`, `cut`, `cp`, `echo`, `eval`, `exit`, `expr`, `grep`, `head`, `pwd`, `read`, `return`, `rm`, `shift`, `tail`, `tee`, `test`, `tr`, `wc`.

Exercice 14

Écrire un script permettant d'afficher à l'écran le contenu d'une série de fichiers dont les noms sont passés en paramètres, et n'utilisant pas de boucle `for`. Le message `<nom fichier> inaccessible` sera affiché si le fichier `<nom fichier>` est inexistant ou ne peut être lu.

L'écriture de ce script nécessite l'utilisation de la commande `shift` et d'une boucle `while`. Il n'est pas récursif.

Exercice 15

On se propose d'écrire un script `inverse.sh` qui réécrit un fichier en inversant l'ordre de ses lignes. Ce script reçoit en paramètre le nom du fichier à inverser. Ce paramètre peut être éventuellement suivi du nom d'un autre fichier. Dans ce cas, l'inversion du fichier initial sera effectuée dans le second fichier (le fichier initial n'étant pas modifié). On pourra suivre, étape par étape, les indications suivantes pour parvenir à l'écriture finale du script :

1). Écrire le script `inverse.sh` qui teste si le nombre de paramètres reçus est bien égal à `1` ou à `2`. Dans le cas contraire, afficher le message suivant :

```
Erreur : inverse.sh nom_fichier1 [nom_fichier2]
```

Produire un code de retour en rapport avec un appel correct ou erroné de la commande.

2). Si le nombre de paramètres est égal à `2`, recopier le fichier correspondant au premier paramètre dans le second et relancer le script `inverse.sh`, avec seulement le second paramètre en paramètre (appel récursif). Afin de tester le bon fonctionnement du script, on affichera le message `Un seul paramètre`, dans le cas où le script n'est lancé qu'avec un seul paramètre. Dans le cas d'un appel avec deux paramètres, ce message ne devra apparaître qu'une fois (après l'appel récursif).

3). Dans le cas où il n'y a qu'un seul nom de fichier en paramètre, récupérer dans la variable `nblignes` le nombre de lignes de ce fichier. Tester le script, en affichant la valeur de `nblignes`.

4). Mettre en place la boucle `while` qui permettra de parcourir le fichier. Pour la tester, on affichera la valeur décrémentée de `nblignes` à chaque parcours de boucle.

5). À chaque parcours de boucle, envoyer les `nblignes` premières lignes du fichier passé en paramètre dans le fichier temporaire `.inv1`, puis concaténer la dernière ligne du fichier `.inv1` à `.inv2`. Vérifier, en consultant le contenu de `.inv2` après exécution du script. Contrôler qu'il ne manque pas de lignes (ni la première ni la dernière). Effacer `.inv2` après chaque test.

6). Recopier `.inv2` dans le fichier passé en paramètre. Effacer les fichiers `.inv1` et `.inv2` Tester le script avec un, puis deux fichiers passés en paramètres.

Exercice 16

Écrire un script réalisant les tâches suivantes :

1). Affichage du menu suivant, en boucle :

```
Vous pouvez taper une commande Unix, ou bien :
fin si vous souhaitez quitter l'application
relancer n si vous souhaitez relancer les n dernières commandes
Votre choix :
```

2). Dans le cas où l'utilisateur tape une commande Unix, lecture de la commande tapée, exécution de cette commande, et affichage à l'écran des résultats produits (si la commande n'est pas tapée correctement, le « résultat » est le message d'erreur).

3). Dans le cas où l'utilisateur tape par exemple `relancer 3`, nouvelle exécution des `3` dernières commandes exécutées et affichage à l'écran de ces commandes et des résultats produits correspondants. Les commandes déjà exécutées seront écrites (en tant que chaînes de caractères) dans le fichier `temp`, de la manière décrite dans l'étape suivante :

4). Écriture, à la fin de `temp`, de la commande exécutée (ou des commandes exécutées, si l'utilisateur a tapé par exemple `relancer 3`), ainsi que des résultats produits, sous la forme :

```
$ <commande n>
<résultats produits par la commande n>
$ <commande n+1>
<résultats produits par la commande n+1>
...
```

5). Enfin, retour au début.

On pourra suivre les indications suivantes :

1). Écrire la fonction `affichage_menu`, qui affiche le menu décrit plus haut.

2). Structurer le script à l'aide d'une boucle `until`.

3). Dans le cas où l'utilisateur tape par exemple `relancer 3`, on écrira dans `temp` les `3` dernières commandes qui y ont été déjà écrites, ainsi que les résultats produits, mais on n'écrira pas dans ce fichier la commande `relancer 3` (c'est-à-dire que `temp` ne doit contenir que des commandes Unix).

4). Écrire la fonction `relancer_n` qui, dans le cas où l'utilisateur tape par exemple `relancer 3`, relance les `3` dernières commandes déjà tapées, réalise les affichages correspondants à l'écran, et met le fichier `temp` à jour.

Remarques :

- On supposera qu'aucune ligne de résultats ne commence par le caractère `$` suivi d'un espace.
- On n'oubliera pas de détruire le fichier `temp` à la fin de l'exécution du script.

SÉANCE 5

Commandes utiles pour cette séance : `basename`, `echo`, `exit`, `expr`, `test`.

Exercice 17

Écrire le script `arbre_base.sh` permettant d'afficher tous les sous-répertoires d'un répertoire donné en paramètre (quel que soit le niveau des sous-répertoires dans l'arborescence). Ce script de base devra afficher le résultat sous la forme suivante :

```
$ arbre_base.sh TEST
FILS1
S_FILS11
S_FILS12
SS_FILS121
S_FILS13
SS_FILS131
SS_FILS132
FILS2
S_FILS21
FILS3
S_FILS31
$
```

Remarques :

- Pour récupérer cette arborescence de test, taper :
  `cp -r users/linfg/linfg0/S3/TEST .`
- Les fichiers de la sous-arborescence ne doivent pas être affichés par ce script (dans l'arborescence à récupérer, le répertoire `FILS3` contient un fichier `toto` qui est un frère de `S_FILS31`).

Exercice 18

Attention : cet exercice est une version un peu plus compliquée de l'exercice précédent. Il est donc conseillé de commencer par terminer l'exercice 1.

Écrire le script `arbre.sh` permettant de dessiner à l'écran la sous-arborescence d'un répertoire passé en paramètre, et ce sous la forme suivante :

```
$ arbre.sh TEST
TEST
+- FILS1
|  +- S_FILS11
|  +- S_FILS12
|  |  +- SS_FILS121
|  +- S_FILS13
|     +- SS_FILS131
|     +- SS_FILS132
+- FILS2
|  +- S_FILS21
+- FILS3
   +- S_FILS31
$
```

Remarque : par rapport à l'exercice 1, il est demandé ici que le nom du répertoire passé en paramètre soit affiché en premier.

On pourra commencer par le premier résultat intermédiaire suivant :

```
$ arbre.sh TEST
TEST
FILS1
S_FILS11
S_FILS12
SS_FILS121
S_FILS13
SS_FILS131
SS_FILS132
FILS2
S_FILS21
FILS3
S_FILS31
$
```

puis continuer avec le deuxième résultat intermédiaire suivant :

```
$ arbre.sh TEST
TEST
+- FILS1
|  +- S_FILS11
|  +- S_FILS12
|  |  +- SS_FILS121
|  +- S_FILS13
|  |  +- SS_FILS131
|  |  +- SS_FILS132
+- FILS2
|  +- S_FILS21
+- FILS3
|  +- S_FILS31
$
```

On conseille de tenir compte des indications suivantes :

- Considérer que le script `arbre.sh` est une version modifiée de `arbre_base.sh` qui peut être appelée soit avec un seul paramètre, soit avec deux paramètres :
  - Lorsque l'utilisateur lance `arbre.sh`, il ne passe qu'un seul paramètre, qui est le nom d'un répertoire.
  - Lors des appels récursifs de `arbre.sh`, deux paramètres lui sont passés. Le second paramètre est une chaîne de caractères (placée entre guillemets) qui doit être affichée devant le nom d'un sous-répertoire. Cette chaîne de caractères est composée d'espaces, de caractères `|` et de caractères `+`
- On a intérêt à gérer un compteur de sous-répertoires. Lorsque la valeur de ce compteur est égale à `1` (le sous-répertoire à afficher est le dernier), une gestion spécifique de l'affichage doit être effectuée.

Attention :

- Tester si un répertoire est accessible en lecture et en exécution, avant d'éventuellement lancer un appel récursif.
- Alors que la commande :
  `echo a  b`
  affiche la chaîne suivante :

`a b`
qui ne contient qu'un seul espace entre les caractères `a` et `b`, chacune des commandes suivantes :
- `echo "a b"`
- `echo 'a b'`

affiche la chaîne :
`a b`
qui contient bien six espaces entre les caractères `a` et `b`.