Université Paul Sabatier **UE : EDINF3A1**
L2 Informatique **Systèmes 1**

Contrôle terminal
Mardi 12 décembre 2017
2h00 - sans document

L'épreuve comporte trois exercices indépendants répartis en trois parties auxquelles vous devez répondre sur trois copies différentes, en n'oubliant pas de mettre votre numéro étudiant, nom et prénom et la partie (1, 2, ou 3) sur chaque copie. Même si vous ne répondez pas à certaines parties, vous devez tout de même rendre la ou les copies correspondantes.

L'écriture des scripts en shell de Bourne sous-entend l'écriture de vérifications appropriées sur les paramètres, ainsi qu'une gestion cohérente des codes de retour. Il est conseillé de commencer par lire l'ensemble de l'énoncé et de tenir compte du barème. La propreté de la copie sera prise en compte, en particulier en ce qui concerne la lisibilité des caractères ' et \.

PARTIE 1

Exercice 1 (6 points)

1. Réaliser chacune des tâches suivantes à l'aide d'une seule ligne de commande – i.e. pas de `;` (point virgule) ni de structure de contrôle de script (`while, if...`) mais branchements (`|`) ou redirections possibles.
   (a) Afficher le nombre d'éléments du répertoire courant (fichiers ordinaires, sous-répertoires ou autres), cachés ou non, dont le nom contient au moins un caractère `point`. Attention un sous-répertoire peut aussi contenir un caractère `point` dans son nom.
   (b) Écrire dans le fichier `tailles.txt` les noms des fichiers ordinaires non cachés du répertoire courant précédés de leur taille en octet. Si le fichier `tailles.txt` existe déjà, son contenu doit être écrasé.
   (c) Effacer tous les fichiers du répertoire `/tmp` dont le nom commence par une majuscule ou le caractère `$`, suivi d'une lettre (majuscule ou minuscule), suivi d'une chaîne quelconque éventuellement vide et se terminant par un espace suivi d'un chiffre.
2. Expliquer de manière précise ce que font les commandes suivantes (l'explication de la commande est plus importante que son résultat) :
   (a) `grep 'z' g.txt >> f.txt`
   (b) `echo "bonjour" | wc -l >> toto`

PARTIE 2

Exercice 2 (8 points)

On se propose d'écrire deux scripts, `pere.sh` et `fils.sh` qui vont afficher à tour de rôle des messages. La syntaxe d'appel de ces deux scripts sera :

```
pere.sh
fils.sh pid_pere
```

La synchronisation utilisera les commandes `trap`, `kill` et `sleep`.

Le script `pere.sh` commence par lancer, en ***tâche de fond***, le script `fils.sh`, puis réalise son traitement qui correspond à l'algorithme suivant :

```
TantQue vrai Faire
    Afficher ("c'est le tour du père");
    Attendre une seconde; # simulation de son traitement
    Afficher ("le père donne le tour au fils");
    Donner le tour au fils;
    Afficher ("le père attend son tour");
    Attendre son tour;
Fin TantQue;
```

Le script `fils.sh` réalise le traitement suivant :

```
TantQue vrai Faire
    Afficher ("le fils attend son tour");
    Attendre son tour;
    Afficher ("c'est le tour du fils");
    Attendre une seconde; # simulation de son traitement
    Afficher ("le fils donne le tour au père");
    Donner le tour au père;
Fin TantQue;
```

1. Programmer les deux fonctions suivantes, utilisées ensuite dans les deux scripts :
   — `attendreTour` qui permet au processus appelant d'attendre qu'il ait reçu le tour (signal `SIGCONT`). Pour attendre, le processus réalisera une boucle dans laquelle il appelle la commande `sleep 1` jusqu'à ce qu'il ait le tour. Le tour est reçu lorsqu'il a reçu le signal `SIGCONT`.
   — `donnerTour pidAQui` qui donne le tour au processus dont le PID est transmis en paramètre. Pour signaler à un processus qu'il reçoit le tour, le signal `SIGCONT` lui est envoyé.

   On rappelle que, lorsqu'un signal est reçu par un processus, un traitement est exécuté par ce processus. Par défaut, ce traitement conduit généralement à la mort du processus, mais que ce comportement peut être modifié en utilisant la commande `trap`.
2. En supposant les deux fonctions précédentes données, programmer les deux scripts `pere.sh` et `fils.sh`.

Une exécution de cette application conduit à un affichage tel que :

```
C'est le tour du pere de pid 3016
Le fils de pid 3017 attend son tour
Le pere de pid 3016 donne le tour au fils de 3017
Le pere de pid 3016 attend son tour
C'est le tour du fils de pid 3017
Le fils de pid 3017 donne le tour au pere de 3016
Le fils de pid 3017 attend son tour
C'est le tour du pere de pid 3016
Le pere de pid 3016 donne le tour au fils de 3017
Le pere de pid 3016 attend son tour
C'est le tour du fils de pid 3017
```

Remarques : $$ est le PID du shell courant et $! est le PID du dernier processus activé en tâche de fond.

PARTIE 3

Exercice 3 (6 points)

Écrire en shell de Bourne le script `afficher.sh` dont la syntaxe d'appel est la suivante :
`afficher.sh -n|-c expr_reg repertoire`
qui affiche sur `stdout` le contenu de tous les fichiers sélectionnés de l'arborescence ayant pour racine `repertoire`, quel que soit le niveau des fichiers dans l'arborescence.

Les options -n et -c permettent de spécifier la méthode de sélection :
— l'option -n affiche le contenu des fichiers dont le nom correspond à l'expression régulière `expr_reg`,
— l'option -c affiche le contenu des fichiers contenant, dans au moins une ligne, une chaîne de caractères correspondant à l'expression régulière `expr_reg`.

L'affichage du contenu de chaque fichier est précédé des informations suivantes : 1) nom du fichier, 2) nombre de lignes du fichier et 3) la taille en octets en respectant strictement le format ci-dessous :

```
**************
*** fichier : ../syst2/tp1b/texte.txt
*** nombre de lignes : 3
*** taille : 15 octets
**************
toto
titi
tata
```

Par exemple, la commande :
`afficher.sh -n '\.txt$' .`
affiche le contenu des fichiers d'extension .txt qui se trouvent dans l'arborescence dont la racine est le répertoire courant. L'affiche suivra le format défini ci-dessus.

Pour chaque fichier, le script doit vérifier les droits d'accès. Si un fichier n'est pas accessible, un message doit être affiché sur `stderr`, mais l'exécution du script ne doit pas être interrompue.

RAPPELS

cat	`cat [fichier]` affiche sur `stdout` le contenu du fichier dont le nom est passé en paramètre (ou de `stdin` par défaut). Avec l'option `-n`, les lignes affichées sont précédées de leur numéro, la première ligne portant le numéro 1.
cut	`cut -f liste -d délimiteur` ne retient sur chaque ligne de `stdin` que les champs spécifiés par `liste` séparés par `délimiteur`. Le numéro du premier champ sur la ligne est 1. L'option `-f1,3-5` conduira la commande à ne garder que les champs 1, 3, 4 et 5 de chaque ligne. L'option `-f4-` gardera tous les champs à partir du quatrième (compris) de chaque ligne.
echo	La commande `echo` affiche sur `stdout` ses paramètres suivis d'un caractère `'\n'`.
exit	La commande `exit [code]` permet de quitter le script en retournant le code `code`. Sans paramètre, le code retourné est le code de retour de la dernière commande précédemment exécutée.
expr	La commande `expr` permet de réaliser des opérations entières grâce aux opérateurs `+`, `-`, `\*`, `/`, `%`. Si l'un des opérandes n'est pas un entier, le code de retour est strictement supérieur à 1. Attention, dans ce cas un message d'erreur est affiché sur `stderr`.
grep	`grep expression [fichier ...]` affiche sur `stdout` les lignes des fichiers passés en paramètres (les lignes de `stdin` par défaut) qui correspondent au modèle décrit par l'expression régulière `expression`. Dans cette dernière, `^` désigne le début de la ligne, `$` la fin de la ligne, `.` un caractère quelconque, `\{n\}` une répétition de `n` fois le caractère qui précède, `*` une répétition (de longueur quelconque éventuellement vide) du caractère qui précède et `\` ń protège ż le caractère qui suit. Le code de retour de `grep` est nul si au moins une occurrence est trouvée. Avec l'option `-n`, chaque ligne qui correspond à l'expression régulière est précédée de son numéro (la première ligne a le numéro 1), suivi du caractère `:` Avec l'option `-c`, la commande `grep` affiche uniquement sur `stdout` le nombre de lignes qui correspondent à l'expression régulière. Avec l'option `-v`, la commande `grep` affiche sur `stdout` les lignes qui ne correspondent pas à l'expression régulière.
kill	La commande `kill -signal pid` envoie le signal `signal` au processus d'identificateur `pid`. `kill -SIGINT 100` envoie le signal `SIGINT` au processus d'identificateur `100`.
ls	La commande `ls REP` affiche sur `stdout` la liste des noms de fichiers et de répertoires contenus dans le répertoire `REP`. La commande `ls -al REP` affiche sur `stdout` des renseignements détaillés concernant le contenu complet du répertoire `REP`. Sans paramètre, c'est le répertoire courant qui est considéré. Sans l'option `-a`, les fichiers cachés sont ignorés. Les renseignements détaillés qui sont affichés sont constitués de neuf champs (la taille en octets est le cinquième champ et le nom du fichier ou du répertoire est le neuvième champ), deux champs successifs étant séparés par une séquence non vide d'espaces.
rm	`rm fichier [...]` permet de supprimer un ou plusieurs fichiers passés en paramètres. `rm -rf désignation_répertoire [...]` supprime les répertoires passés en paramètres ainsi que leur contenu (fichiers et sous-répertoires).
test	À la suite de la commande `test`, peuvent apparaître les opérateurs logiques `-a`, `-o`, `!`, les comparateurs d'entiers `-eq`, `-ne`, `-gt`, `-ge`, `-lt`, `-le`, les comparateurs de chaînes de caractères `=`, `!=`, les opérateurs sur l'existence d'un fichier ou d'un répertoire, un fichier ou un répertoire `-e`,`-f`, `-d`, `-r`, `-w`, `-x`.
tr	`tr -d chaîne` supprime toutes les occurrences des caractères composant `chaîne`. `tr -s ch` élimine les répétitions des caractères composant `ch` en n'en laissant qu'un à la fois.
trap	`trap commande signal` permet au processus courant d'exécuter la commande `commande` à l'arrivée du signal `signal`.
wc	La commande `wc -l` provoque l'affichage sur `stdout` du nombre de lignes provenant de `stdin`. Ce nombre est précédé d'un ou de plusieurs espaces.

Remarque : les informations données ci-dessus en guise de rappel n'impliquent pas que les commandes indiquées doivent obligatoirement apparaître dans les réponses aux exercices, ni que ce sont les seules qui doivent être utilisées.