Projet Langage de Scripts – SHELL : « Trier en Shell »

LMI3 Informatique

1 Objectifs

L'objectif de ce projet est de réaliser en *shell* (`/bin/bash`) un programme qui trie les entrées d'un répertoire suivant différents critères.

2 Description

La syntaxe de votre commande sera : `trishell [-R] [-d] [-nsmletpg] rep`

Les différentes options sont les suivantes :

- `-R` : tri le contenu de l'arborescence débutant au répertoire `rep`. Dans ce cas on triera par rapport aux noms des entrées mais on affichera le chemin d'accès ;
- `-d` : tri dans l'ordre décroissant, par défaut le tri est effectué dans l'ordre croissant ;
- `-nsdletpg` : permet de spécifier le critère de tri utilisé. Ces critères peuvent être combinés, dans ce cas si deux fichiers sont identiques pour le premier critère, le second critère les départegera et ainsi de suite.
  - `-n` : tri suivant le nom des entrées ;
  - `-s` : tri suivant la taille des entrées ;
  - `-m` : tri suivant la date de dernière modification des entrées ;
  - `-l` : tri suivant le nombre de lignes des entrées ;
  - `-e` : tri suivant l'extension des entrées (caractères se trouvant après le dernier point du nom de l'entrée ;
  - `-t` : tri suivant le type de fichier (ordre : répertoire, fichier, liens, fichier spécial de type bloc, fichier spécial de type caractère, tube nommé, socket) ;
  - `p` : tri suivant le nom du propriétaire de l'entrée ;
  - `g` : tri suivant le groupe du propriétaire de l'entrée.

Exemple : `trishell -R -d -pse /home` trie, par ordre décroissant, l'arborescence débutant à `/home` en fonction des propriétaires des entrées comme premier critère, de la taille des entrées comme second critère et de l'extension des entrées comme dernier critère.

3 Recommandations

3.1 Comparaison de chaînes de caractères

Pour comparer deux chaînes de caractères en *shell*, il est possible d'utiliser la commande `test` avec les opérateurs < et > bien que ces derniers ne soient pas référencés dans la page *man*.

Exemple :

```
#!/bin/bash
chaine1="foo"
chaine2="bar"
if (test $chaine1 \< $chaine2)
then
   echo OK
else
   echo KO
fi
if (test $chaine1 \> $chaine2)
then
   echo OK
else
   echo KO
fi
```

Le premier test produira l'affichage de `KO` et le second celui de `OK`. Remarquez que l'utilisation du *backslash* (\) est obligatoire sinon le *shell* interprète les caractères < et > comme des caractères de redirections.

3.2 Méthode de tri

Vous êtes libres d'utiliser la méthode de tri de votre choix. Cependant, les tris en $O(n \log n)$ seront préférés aux tris quadratiques. Un bonus sera donc attribué aux programmes ayant utilisé une méthode efficace. Ceci étant dit, il vaut mieux un tri quadratique qui marche qu'un tri efficace inopérationnel !!!

3.3 Fonctions en *shell*

Il est possible de programmer des fonctions en *shell*. Une fonction du *shell* mémorise une série de commandes pour permettre une exécution ulterieure. Les déclarations des fonctions se font au début du programme et respectent la syntaxe suivante :

```
function  nom()
{
   instructions composant la fonction
}
```

Les fonctions sont exécutées dans le contexte de l'interpréteur en cours. On ne crée pas de nouveau processus pour interpréter une fonction, contrairement à l'exécution d'un script.

Les arguments d'une fonction sont placés dans les paramètres positionnels (`$1, $2, $3, ...`) durant son exécution. Le paramètre spécial `$#` est mis a jour. Le paramètre positionnel `$0` n'est pas modifié. À la fin de l'exécution de la fonction, les paramètres positionnels sont rétablis automatiquement.

Il est possible de déclarer des variables locales à une fonction qui ne conserveront pas leur valeur et qui ne seront donc pas visibles à l'extérieur de la fonction. Les variables locales d'une fonction peuvent être déclarées en utilisant la commande interne `local`. Autrement, les variables et leurs valeurs sont partagées entre la fonction et son appelant. C'est-à-dire que toutes les variables du script sont visibles dans les fonctions et toutes les variables non locales déclarées dans la fonction sont visibles à l'extérieur de la fonction.

Si la commande interne `return` est exécutée dans une fonction, celle-ci se termine et l'exécution reprend avec la commande suivant l'appel de fonction. Quand une fonction se termine, les paramètres positionnels et le paramètre spécial `$#` reprennent les valeurs qu'ils avaient avant l'appel de fonction.

Les fonctions peuvent être récursives. Aucune limite n'est imposée quant au nombre d'appels récursifs.

La valeur de retour de la fonction est celle de la dernière instruction exécutée.

L'appel d'une fonction se fait en indiquant son nom suivi de ses paramètres séparés.

Exemple :

```
#!/bin/bash

function calc()
{
  local op

  op=$2
  if (test "$op" = "x")
  then
     res=‘expr $1 \* $3‘
  else
     res=‘expr $1 $op $3‘
  fi
}

if (test $# -ne 3)
then
  echo "Erreur 3 paramètres"
else
  case $2 in
  /) if (test $3 -eq 0)
     then
        echo "Division par zéro"
        exit 1
     fi
     calc $1 $2 $3
     echo $res
     ;;
  +|-|x)
     calc $1 $2 $3
     echo $res
     ;;
  *)  echo "Opérateur inconnu"
     ;;
  esac
fi
```

4 Informations pratiques

Ce projet est à réaliser en binôme. Votre programme devra être envoyé par mail avant le mardi 4 décembre 2018 minuit à l'adresse `chevallier@cril.fr`
Un mail d'accusé réception vous sera adressé en retour.
Le sujet du mail devra être :

`[SHELL: projet tri] nom_binome1-prenom_binome1, nom_binome2-prenom_binome2`

Le corps du mail contiendra le détail des fonctionnalités réalisées, de celles non réalisées et pourquoi, ainsi que l'ensemble des explications que vous jugerez nécessaires à l'évaluation de votre travail. Enfin vous indiquerez la part (en pourcentage) de travail de chacun des binômes sur le projet (tout projet où cette proportion n'est pas indiquée ne sera pas corrigé et vaudra donc 0, cette information n'influera pas ou peu sur la note finale du binôme).

ATTENTION :
Pour ce projet, les commandes `sort` et `ls`, ainsi que tout autre commande réalisant automatiquement des tris sont interdites. L'utilisation des fonctionalités tableaux offertes par `bash` (et non par `sh`) est également proscrite.