TP1 : Arborescence, droits d'accès, interpréteur de commandes et shell

CSH : Initiation au C et au shell

Première année

1 Préliminaires

Au cours des séances de TP, vous allez être confrontés à de nombreux messages d'erreurs, prévus ou non par l'énoncé. Constituez-vous dès aujourd'hui une liste de ces messages, à compléter au fil des séances. Pour chaque message, notez également sa signification et comment corriger ou contourner le problème pour gagner du temps à l'avenir.

Conventions d'écriture utilisées

- les noms des commandes Unix sont écrits en caractères `gras` ;
- les noms de fichiers et de répertoires sont écrits en caractères ***gras et italiques*** ;
- les lignes correspondant à des entrées et des sorties à l'écran sont écrites en police de caractères `courier` ;
- les commandes à taper sont écrites `comme ceci` ;

<TAB>

- désigne la touche de tabulation, et <ctrl-d> les touches «Control» et «d» pressées en même temps.

D'autre part, on écrira « placez-vous dans le répertoire ***REP***» ou «allez dans le répertoire ***REP***», pour dire «votre répertoire de travail doit être le répertoire ***REP***». C'est une convention de langage. Pour alléger le texte, on utilisera aussi la notation ***[REP]*** en tête d'alinéa ou de paragraphe, avec la même signification.

Environnement de travail Dans le cadre de ce module (et du module Informatique de Base – IB1), vous travaillerez avec linux. L'objectif de ce premier TP est de vous familiariser avec cet environnement si ce n'est pas encore fait. Il est fortement conseillé d'utiliser l'éditeur `emacs`. La première étape est de vous connecter («login») à l'environnement.

Avant de commencer : exécutez avant toute chose les commandes suivantes.

- `cp /home/depot/1A/CSH/.cshrc .` (le point final fait partie de la commande).
- `source .cshrc`

2 Répertoires, fichiers, fichiers de commandes

Votre répertoire de travail, au début de la session, est votre répertoire personnel (*home directory*). Vérifiez-le à l'aide de la commande `pwd` (**p**rint **w**orking **d**irectory). Consultez le contenu de ce répertoire, à l'aide de la commande `ls` (**l**i**s**t).

Créez, dans votre répertoire personnel, le répertoire ***CSH*** (par la commande `mkdir CSH`). Tous les répertoires et fichiers créés au cours de cet enseignement seront placés dans ce répertoire, nommé ***RÉPERTOIRE PRINCIPAL***.

[CSH] Activez une fenêtre de l'éditeur par la commande `emacs &` et tapez le texte suivant dans cette fenêtre :

```
#!/bin/sh
cd $HOME/CSH
mkdir TP1
cd TP1
cp /home/depot/1A/CSH/TP1/*
echo le repertoire courant est
pwd
echo il contient
ls -l
```

Enregistrez ce texte dans un fichier de nom ***copiedir.sh*** dans votre répertoire principal (raccourci : <ctrl-x> <ctrl-s>). Revenez à l'environnement Unix et vérifiez que ce fichier est bien créé. Affichez son contenu (`less copiedir.sh`) et vérifiez que vous n'avez pas fait d'erreur de saisie (faute de frappe).

Exécutez la commande `ls -l copiedir.sh`. Parmi les différents champs du résultat, on trouve le nom du fichier, sa date de création, sa taille et ses droits d'accès. Notez ces quatre informations.

Le fichier ***copiedir.sh*** est un *fichier de commande* (ou *shell-script*). On peut exécuter les commandes présentes dans un tel fichier. Exécutez la commande `./copiedir.sh` (le nom du fichier). Notez le message d'erreur obtenu.
Ce message signifie que vous n'avez pas le droit d'*exécuter* ce fichier. Il faut donc modifier les droits d'accès en utilisant la commande `chmod u+x copiedir.sh`. Exécutez cette commande.

Quelle commande vous permet de vérifier que les droits d'accès sur ce fichier ont changé ? Faites-le.

Exécutez maintenant à nouveau la commande `./copiedir.sh`. Lisez attentivement le texte des messages qui sont affichés. Notez le nouveau message d'erreur. Il signifie qu'il y a une erreur dans la ligne de commande qui commence par `cp`. Il manque l'indispensable deuxième argument de cette commande : il indique où il faut copier le (ou les) fichier(s) indiqué(s) en premier argument. Dans le fichier ***copiedir.sh***, modifiez la ligne erronée ainsi :

```
1 cp /home/depot/1A/CSH/TP1/* .
```

Après avoir ainsi corrigé le contenu du fichier ***copiedir.sh***, exécutez à nouveau la commande `./copiedir.sh`. Quel a été l'effet de cette commande ? Que vous indique le message d'erreur obtenu ?

Quel est maintenant votre répertoire de travail ? Exécutez la commande `pwd`. Est-ce conforme à ce que vous attendiez ?

Pour tout ce qu'on vous demande dans le paragraphe suivant, placez-vous dans votre répertoire ***TP1*** et effectuez les opérations demandées SANS CHANGER DE RÉPERTOIRE DE TRAVAIL.

[TP1] Créez dans votre répertoire personnel un répertoire ***TP1bis*** (`mkdir ../TP1bis`) et copiez dans ce répertoire l'ensemble des fichiers présents dans votre répertoire ***TP1***. Comparez les dates de création de ces fichiers avec celles des fichiers du répertoire ***TP1***. Effacez un par un tous les fichiers du répertoire ***TP1bis*** (avec la commande `rm`), puis le répertoire ***TP1bis*** lui-même (commande `rmdir`).

3 Problème de droits d'accès

[CSH] Pour cette question, travaillez avec votre voisin ou quelqu'un qui en est au même point que vous, et appellez-le COLLEGUE. Vérifiez qu'il n'est pas possible de copier le fichier ***copiedir.sh*** dans le répertoire personnel de COLLEGUE. Que se passe-t-il si vous essayez ?

Modifiez ***copiedir.sh*** en remplaçant la ligne `cd $HOME/CSH` par `cd $HOME/CSH2`. Échangez votre session avec COLLEGUE, c'est-à-dire échangez votre place avec la personne à coté de vous. Créez dans son répertoire personnel le répertoire ***CSH2***.

[~COLLEGUE/CSH2 en tant que COLLEGUE] Copiez le fichier ***copiedir.sh*** qui est dans VOTRE répertoire ***CSH*** dans le répertoire ***CSH2*** de COLLEGUE. Comment expliquez-vous que ce soit possible, cette fois-ci ? Exécutez alors la commande `./copiedir.sh`. Pourquoi n'est-il pas nécessaire d'utiliser la commande `chmod` au préalable ?

Retournez dans le répertoire personnel de COLLEGUE, effacez le répertoire ***CSH2*** à l'aide de la commande `rm -rf CSH2/` puis retournez à votre session.

[CSH] Modifiez à nouveau ***copiedir.sh*** en remplaçant la ligne `cd $HOME/CSH2` par `cd $HOME/CSH`.

4 Des ennuis fréquents et comment y remédier

[TP1] Les commandes suivantes vont provoquer des erreurs. Notez soigneusement chaque message d'erreur, à quelle erreur il correspond et comment corriger.

`copiedir.sh`	on n'est pas dans le bon répertoire
`../copidir.sh`	une «faute d'orthographe» dans le nom de la commande
`cp fich1`	mauvais nombre d'arguments
`cp ../copidir.sh .`	une «faute d'orthographe» dans un nom de fichier
`cp ../TP2/copiedir.sh .`	une «faute d'orthographe» dans un chemin d'accès (*path*). Comprenez-vous pourquoi le message est le même que dans le cas précédent ?
`cp ../copiedir.sh TP1/cop.sh`	une «faute d'orthographe» dans un chemin d'accès (*path*) Comprenez-vous pourquoi le message n'est pas le même que dans les cas précédents ?
`cp ../copiedir.sh .`	la commande correcte, enfin.

Supprimez le fichier ***copiedir.sh*** du répertoire principal (votre $HOME/CSH).

5 Pour se simplifier la vie...

Les fautes de frappe et les erreurs de chemin d'accès (le répertoire de travail n'est pas celui que l'on croit, on ne connaît pas le nom exact des répertoires traversés, etc.) sont monnaie courante, même pour des utilisateurs expérimentés. Voici quelques petites choses faciles à utiliser pour les éviter.

[CSH]

1. Tapez successivement sur les touches : `./copied`<TAB> Que s'affiche-t-il à l'écran ?
   Essayez de taper moins de caractères avant <TAB>. Combien sont nécessaires ?
   Dupliquez ***copiedir.sh*** en un fichier nommé ***copain*** (`cp copiedir.sh copain`), et refaites l'expérience ci-dessus. Combien de caractères sont maintenant nécessaires ?
   Tapez successivement sur les touches : `./cop`<ctrl-d> Que voyez-vous ?
   Et que voyez-vous si vous tapez seulement `./co`<ctrl-d> ?
   <ctrl-d> est très utile également pour se déplacer dans l'arborescence des répertoires et fichiers. Tapez par exemple :
   `cd /ho`<TAB><ctrl-d>`dep`<TAB>`/1A/CS`<TAB>`/TP`<ctrl-d>`1`<ctrl-d>
2. Certaines commandes sont longues et fastidieuses. On retrouve les commandes précédentes avec les touches «flèche vers le haut» et «flèche vers le bas». Quelle était la quatrième précédente commande que vous aviez exécutée ?
   Pour retrouver une commande plus «ancienne», utilisez la commande **history**. Pour exécuter la commande dont le numéro est n, tapez la commande `!`n. Pour exécuter la quatrième commande précédente, tapez la commande `!`-4. Pour exécuter la quatrième commande de la session courante, tapez la commande `!4`.
3. Utilisation du caractère `*` (dans les commandes `cp` et `mv`, par exemple).
   Dans la ligne suivante de ***copiedir.sh***, l'étoile représente tous les fichiers du répertoire ***/home/depot/1A/CSH/TP1*** : `cp /home/depot/1A/CSH/TP1/* .`
   Cette ligne de commande permet donc de copier, en une seule fois, tous les fichiers présents dans le répertoire ***1A/CSH/TP1*** dans le répertoire de travail.

Utilisez ces facilités pour refaire plus rapidement l'expérience demandée précédemment (toujours sans changer de répertoire de travail, cependant) :

[TP1] Créez dans le répertoire principal ***CSH*** un répertoire ***TP1bis*** et copiez dans ce répertoire l'ensemble des fichiers présents dans votre répertoire ***TP1*** ; comparez les dates de création de ces fichiers avec celles des fichiers du répertoire ***TP1*** ; effacez tous les fichiers du répertoire ***TP1bis***, puis le répertoire ***TP1bis*** lui-même.

6 Exercices divers

▷ **Question 1.** Le répertoire courant est votre répertoire principal. Donnez une suite de commandes permettant d'échanger les contenus des fichiers nommés ***fich1*** et ***x2.c***,

1. s'ils sont tous les deux dans le répertoire ***TP1***.
2. si ***fich1*** est dans votre répertoire principal et ***x2.c*** dans votre répertoire ***TP1***.

▷ **Question 2.** L'utilisateur `alfred` tape les commandes suivantes dans une fenêtre Unix :

```
{alfred} 42 > ls
TP1   TP2   essai  copiedir.sh
{alfred} 43 > cp  essai
{alfred} 44 > cpessai  TP1/truc
{alfred} 45 > cp  TP1/fich1  ../TP2/fich1
```

Pour chaque commande, dites si un message d'erreur est affiché et si oui, indiquez lequel et sa signification.

7 Debugger les scripts shells

Lorsque l'on écrit des programmes, on est souvent amené à corriger des erreurs qui s'y sont glissés. Les scripts ne font pas exception à la règle, malheureusement. Contrairement au C ou au Java, il n'existe pas d'outil spécifique pour debugger des scripts shell. Il faut donc suivre une approche incrémentale en testant son script régulièrement en cours de réalisation.

Vous trouverez dans `/home/depot/1A/CSH/TP1/nombre_mystere.sh` un petit script qui présente plusieurs problèmes, et qu'il convient de corriger maintenant. Il contient certaines constructions que nous n'avons pas encore vu, mais les commentaires devraient suffire à les comprendre. Les problèmes commencent lorsque l'on exécute le script :

```
[quinson@neptune TP1]$ ./nombre_mystere.sh
./nombre_mystere.sh: line 20: unexpected EOF while looking for matching `''
./nombre_mystere.sh: line 29: syntax error: unexpected end of file
```

Cela signifie qu'il manque des guillemets fermants quelque part. Ce message d'erreur est le cauchemar principal de tout codeur en shell, puisque localiser l'erreur peut s'avérer assez difficile. Ici, par exemple, la ligne 20 (`echo "Non, c'est trop grand"`) comme la ligne 29 (`exit 0`) sont parfaitement valides. Seule certitude : l'erreur se trouve avant la ligne 20, et l'interpréteur «déraille» à partir de là.

La première approche consiste à ouvrir le script dans emacs, et faire confiance à son module de colorisation. Essayez. Après la ligne coupable, tout le texte est en bordeaux, sauf les chaînes (qui devraient l'être), qui sont elles noires[1]. Corrigez le problème et relancez le script.

Aïe. Cette fois, c'est une parenthèse qui manque. La bonne nouvelle est que le message d'erreur indique le numéro de ligne correct (avant d'autres messages sans intérêt). La leçon ici est que lorsqu'on débuggue un programme, il ne faut lire que le tout premier message d'erreur, et ne pas passer à la suite tant qu'il n'est pas corrigé. En effet, après la première erreur de syntaxe, le jugement de l'interpréteur n'est plus très fiable. Cette leçon n'est d'ailleurs pas limitée au shell : la même chose est vraie en C ou en Java.

Une fois cette dernière erreur de syntaxe corrigée, nous ne sommes pas sortie d'affaire pour autant.

```
Votre proposition : 50
Non, c'est trop petit
Votre proposition : 100
Non, c'est trop petit [Ctrl-C]
```

On doit deviner un nombre compris dans [0 ;100], et 100 est une proposition trop petite...Pour trouver cette erreur de logique, il vous faudra sans doute ajouter un certain nombre de `echo` pour suivre l'évolution de chacune des variables.

Une autre solution consiste à invoquer le shell avec l'option -x pour lui demander à afficher chacune de ses actions et leurs résultats. On peut le faire soit en ligne de commande (`bash -x ./nombre_mystere.sh`), soit en ajoutant `set -x` dans le script, soit en utilisant la toute première ligne du script (`#! /bin/bash -x`). `set +x` dans le script arrête les affichages avant la fin.

Bon debuggage...

[1] Cette approche ne fonctionne pas toujours ; il faut utiliser un `set -x` ou autre chose quand la colorisation se trompe.