Université Grenoble Alpes
U.F.R. Sciences de l'Homme et de la Société
Master MIASHS DCISS

TP2 : Langage de commande Shell d'UNIX

Fichiers de commande

1 Rappels

Script

Les scripts sont des programmes qui permettent d'automatiser certaines tâches répétitives telles que l'administration ou les sauvegardes de fichiers. Ce sont des fichiers de texte en général sans extension ; ils ont parfois l'extension `.sh` ou `.bash`. Ils contiennent les mêmes commandes que celles exécutées en ligne de commande, plus éventuellement des structures itératives (`for`, `while`, `until`) ou conditionnelles (`if`) et des appels à des fonctions. Un script contient en général une commande par ligne. Il peut être paramétré, les paramètres sont désignés par `$1, $2, … $9`.

Exécuter un script shell

Pour exécuter un script shell, il suffit de taper `bash` suivi du nom de ce fichier dans une fenêtre de terminal. L'interprète exécute les commandes de manière séquentielle, dans l'ordre où elles apparaissent dans le fichier. Si l'on donne les droits d'exécution au fichier et que le fichier est place dans un répertoire défini dans la variable `PATH`, il suffit de taper le nom du fichier.

La variable PATH

Pour pouvoir exécuter un programme, le système d'exploitation doit pouvoir retrouver le fichier correspondant à cette commande. C'est la variable d'environnement `PATH` qui contient la liste des répertoires que l'interprète explore pour trouver un programme à exécuter. Vos scripts seront donc placés dans le répertoire `$HOME/bin` (voir TP n°1).

2 Avant de programmer

- Vérifiez que le répertoire `$HOME/bin` existe et qu'il apparaît dans votre variable `PATH`.
- Conseil : utilisez un éditeur (graphique) de texte `scite` pour écrire vos scripts ; sélectionnez **Shell** dans le menu Language pour avoir la coloration syntaxique du langage shell et dans le menu View, sélectionnez **Line numbers** pour voir les numéros de ligne, ce qui peut être utile lors de la mise au point de vos programmes.
  Si vous ne pouvez pas lancer d'applications graphiques, vous pouvez utiliser l'éditeur de texte **nano**
- Evitez d'utiliser les caractères accentués qui sont codés différemment dans les éditeurs sous Ubuntu et dans les fenêtres de terminal.
- Pensez à utiliser votre cours et l'aide en ligne pour la syntaxe des commandes.

3 Scripts à écrire pendant la séance de TD

3.1 Tester les paramètres passés à l'exécution d'un script shell

1. Dans le répertoire `$HOME/bin`, créer le fichier de commandes `testparam` avec le texte ci-contre.
2. Que représentent `$1` et `$*` ?
3. Que fait `[ $1 = "--help" ]` ?
4. Tester ce script en passant 1, puis 2 paramètres, puis le paramètre `–help`

```
#!/bin/bash
if [ $1 = "--help" ]
then
   echo "usage : testparm parametres"
else
   echo "Les parametres sont : $*"
fi
```

5. Que se passe-t-il si l'on teste le script sans passer de paramètre ?
6. Changer `[ $1 = "--help" ]` par `[ "$1" = "--help" ]` . Le script fonctionne-t-il correctement maintenant ? Pourquoi ?
7. Modifier le script pour que le message « `usage : testparm parametres` » soit également affiché lorsque aucun paramètre n'est donné

3.2 Tester l'existence de fichier et/ou répertoire

```
#!/bin/bash
if test $# -eq 0 -o "$1" = "--help"; then
   echo "usage : testfic nomfichier"
elif [ -f $1 ] ; then
   echo "le fichier $1 existe"
elif [ -d $1 ] ; then
   echo "le repertoire $1 existe"
else
   echo $1 est absent !
fi
```

1. Dans le répertoire `$HOME/bin`, créer un fichier de commandes `testfic` qui reprend l'exemple ci-contre.
2. Tester cette commande pour un fichier et un répertoire existants/non existants.
3. Commenter le code :
   - Que représentent le `$#` ?
   - Que fait `if test $# -eq 0 –o $1 = "--help"` ?
   - Que représente `[ –f $1 ]` ?
4. Afficher « l'aide » de la commande (en lançant le programme avec l'option correspondante)
5. Modifiez le script `testfic` de façon à ce qu'il permette de tester l'existence de plusieurs fichiers ou répertoires (nombre indéfini de paramètres) en utilisant la commande `shift` et une itération (voir l'exemple donné en cours).

3.3 Sauvegarder des fichiers et des répertoires

Les scripts shell peuvent être utiles pour la réalisation de sauvegardes automatiques.

- Ecrire la commande `sauver` qui copie des fichiers passés en paramètres dans le répertoire `sauvegarde` du répertoire courant (à créer s'il n'existe pas).
  - Afficher un message d'erreur si un nom correspond à un répertoire ou si un fichier est absent
- Ecrire un script `datedujour` qui affiche :

  `Nous sommes le Xeme jour du Yeme mois de l'annee Z.`

Pour cela, reprendre le script donné en cours qui afficher la date sous cette forme et le modifier de sorte que les zéros non significatifs n'apparaissent pas ; on pourra aussi tenir compte du 1 et afficher 1er au lieu de 1eme

> Pour extraire une sous-chaine d'une variable, on utilise la syntaxe suivante :
> `${nomvar:d:lg}`
> `nomvar` est le nom la variable
> `d` est l'indice de début de la sous-chaine (0 est l'indice du premier caractère)
> `lg` est la longueur de la sous-chaine
> exemple : si la variable `nomvar=tralala`, alors `${nomvar:1:3}` correspond à la chaîne `"ral"`

4 Lecture de fichiers, analyses de chaînes, calculs

- *Pour chacune de vos commandes (scripts), les 3 points suivants doivent être respectés :*
  - *Votre code doit être commenté et chaque commande doit avoir un en-tête précisant son rôle et ses paramètres.*
  - *L'utilisateur doit pouvoir obtenir l'aide des commandes en tapant :* `nomcommande --help`
  - *Si l'utilisateur appelle la fonction avec de mauvais paramètres, un message d'erreur et d'aide approprié doit s'afficher.*

Les questions suivantes concernent la gestion d'une épreuve de lancé de Javelot. Durant cette compétition, chaque athlète a eu 3 essais.

4.1 – Écrire la commande `maximum` qui calcule dans la variable nommée `maxi` et affiche le maximum d'une liste d'entiers passée en paramètres. Le nombre de paramètres n'est pas fixé.

Par exemple, la commande « `maximum 1 2 3 4 3 2 1` » a pour effet d'afficher et de placer dans la variable `maxi` la valeur 4 car il s'agit du maximum des paramètres donnés dans l'exemple.

La commande `maximum` vérifiera la présence d'au moins un paramètre et affichera un message d'aide dans le cas contraire.

4.2 – Écrire la commande `categorie` qui calcule, stocke dans la variable `categ` et affiche la catégorie d'âge d'un athlète.

Par exemple, la commande « `categorie 1984` » a pour effet d'afficher et de placer dans la variable `categ` la valeur `senior`.

En athlétisme, les catégories d'âge de la saison 2016/2017 sont données par le tableau suivant :

CATEGORIE	**ANNEE DE NAISSANCE**	**CATEGORIE**	**ANNEE DE NAISSANCE**
Masters	1976 et avant	Minimes	2001 et 2002
Seniors	1977 à 1993	Benjamins	2003 et 2004
Espoirs	1994 à 1996	Poussins	2005 et 2006
Juniors	1997 et 1998	École d'Athlétisme	2007 à 2009
Cadets	1999 et 2000	Baby Athlé	2010 et après

4.3 – Écrire la commande `afficherresultat` qui affiche la catégorie et le meilleur lancé d'un athlète donné.

Par exemple, la commande « `afficherresultat Martin Galle 12-2-1991 94 87 93` » a pour effet d'afficher : `Le meilleur lancé du senior Martin Galle est de 94 m`

La commande `afficherresultat` doit comporter 6 paramètres : le prénom et le nom de l'athlète, sa date de naissance (au format j-m-aaaa) et les 3 essais.

La commande `afficherresultat` pourra appeler les commandes `maximum` et `categorie`.

4.4 – On dispose d'un fichier de texte appelé `athletes.txt` et contenant une ligne pour chaque athlète avec le format suivant :

`Nom:Prenom:DatedeNaissance:essai1:essai2:essai3`

Exemple de fichier `athletes.txt` :

```
#Fichiers des athletes
#Nom:Prenom:dateNaissance:essai1:essai2:essai3
Galle:Martin:12-2-1991:94:87:93
Honnete:Marie:14-9-1982:82:91:93
Pleur:Jean:13-3-1976:93:79:72
Prane:Dolly:19-11-1999:41:57:59
Mieu:Jeff-Edmond:22-12-2004:89:95:91
```

Ecrire la commande `calculerresultats` qui affiche, pour chaque athlète d'un fichier donné, sa catégorie d'âge et son meilleur lancé. Le fichier contenant les athlètes et leurs performances au javelot est un paramètre de la commande.

Conseils :

1) Pensez à :

- Commenter votre code,
- Testez vos scripts séparément,

- Gérez les erreurs et les aides en ligne.

2) Si un script ne fonctionne pas, faites des traces en affichant les valeurs des variables mises en jeu. Ainsi vous trouverez plus facilement une erreur.