Programmation shell scripts

semaine 2

Structure d’un script

1. spécifier l'interpréteur
2. vérifier la syntaxe
3. signaler le lancement
4. effectuer la/les tâches
5. nettoyer l’environnement
6. signaler la fin de l'exécution

```
#!/bin/bash
[ $# -ne 1 ] && echo "Usage : `basename $0` int" 1>&2 && exit 1
echo debut du travail
for ((i=0; i<$1; i++)); do
    for ((j=0; j<$1; j++)); do
        echo -n '*'
    done
    echo -en '\n'
done
echo fin

# N.B : les variables spéciales seront étudiées plus tard dans ce cours
# N.B : ces étapes ne sont pas toutes obligatoires
```

Ligne shebang

Pensez à insérer la ligne shebang en début de script pour indiquer le nom de l'interpréteur à employer. En général on appellera :

- #!/bin/sh
- #!/bin/bash

Selon le contexte d'utilisation, ou sur certains systèmes (Solaris par exemple), on fera appel à d'autres shells:

- #!/bin/ksh
- #!/bin/zsh

Commentaires

```
# une ligne de commentaire
echo bonjour # ou après une commande
: pseudo commentaire, toujours vrai
```

Les commentaires sont indispensables pour assurer une lisibilité correcte d'un script shell. La syntaxe du shell est parfois complexe et les lignes de commande incluant des appels à grep , sed ou awk se révèlent souvent de véritables défis pour le lecteur, même expérimenté.

Il est donc important de bien commenter un script, en spécifiant le but/rôle d'un ensemble de lignes.

Cela simplifie nettement la relecture ultérieure et les opérations de maintenance sur votre programme.

Commentaires : entête

Pour faciliter la maintenance d'un script, il est intéressant d'insérer quelques lignes de commentaire formant un en-tête, contenant :

- Le nom du script
- Quelques lignes décrivant le rôle et l'utilisation du script
- Le nom de l'auteur et son adresse e-mail, surtout si le programme est destiné à être diffusé
- Un historique des changements

```
###################################
# monscript.sh
# Permet de faire une synthèse des
# différents fichiers log du serveur Apache

# Auteur : Prénom NOM <prenom.nom@hote.com>

# Historique :
# 18/01/2017 (1.3) :
# correction bug sur longueur des champs
# 14/01/2017 (1.2) :
# ajout option -f
# 09/01/2017 (1.1) :
# inversion de l'option -i
# 07/01/2017 (1.0) :
# écriture initiale
###################################
# VERSION=1.3
```

Indentation

Les programmeurs débutants ont parfois du mal à savoir comment décaler leurs lignes de code de la marge.
Il s'agit pourtant d'une très bonne habitude à prendre, qui devient vite un automatisme et améliore la lisibilité du script.
L'organisation logique du script doit apparaître au premier regard ; le début et la fin de chaque structure de contrôle (boucle, condition, fonction...) étant mis en relief par le décalage du code contenu dans la structure.

Affichage des messages

- Nous avons déjà largement utilisé la commande echo qui permet à un script d’envoyer un message sur la sortie standard. C’est une commande interne du shell, mais il en existe souvent un équivalent sous forme d'un exécutable binaire /bin/echo . Les principales options reconnues par la version Gnu (Linux) de “echo” et de son implémentation interne dans Bash sont :
  - -n : Ne pas effectuer de saut de ligne après l’affichage. Cela permet de juxtaposer des messages lors d’appels successifs, ou d’afficher un symbole invitant l’utilisateur à une saisie.
  - -e : interpréter les caractères spéciaux (\n,\r,\\,...)

Affichage avec “echo” : caractères spéciaux

- \\ : Affichage du caractère \
- \XXX : Affichage du caractère dont le code Ascii est XXX exprimé en octal
- \a : Beep
- \c : Éliminer le saut de ligne final (comme l’option -n)
- \n : Saut de ligne
- \r : Retour chariot en début de ligne
- \b : Retour en arrière d’un caractère
- \t : Tabulation horizontale
- \v : Tabulation verticale

Affichage formaté : printf

Permet d'afficher des données formatées. Cette commande est une sorte de "echo" nettement amélioré proposant des formats pour afficher les nombres réels. Les programmeurs C reconnaîtront une implémentation en ligne de commande de la célèbre fonction de la bibliothèque stdio.h

- $ printf chaine arg1 arg2 arg3…

Exemples de formats (voir tableau) :

%20s	Affichage d'une chaine (string) sur 20 positions avec cadrage à droite
%-20s	Affichage d'une chaine (string) sur 20 positions avec cadrage à gauche
%3d	Affichage d'un entier (décimal) sur 3 positions avec cadrage à droite
%03d	Affichage d'un entier (décimal) sur 3 positions avec cadrage à droite et complété avec des 0 à gauche
%-3d	Affichage d'un entier (décimal) sur 3 positions avec cadrage à gauche
%+3d	Affichage d'un entier (décimal) sur 3 positions avec cadrage à droite et affichage systématique du signe
%10.2f	Affichage d'un nombre flottant sur 10 positions dont 2 décimales
%+010.2f	Affichage d'un nombre flottant sur 10 positions dont 2 décimales, complété par des 0 à gauche, avec cadrage à droite et affichage systématique du signe

Exécution d’un shell script

- Par invocation de l'interpréteur :
  - $ bash essai.sh
  - $. essai.sh
  - $ source essai.sh
- Par appel direct :
  - $ essai
    - Déplacer le script dans l’un des répertoires de PATH
    - Ajouter le répertoir du script à la variable PATH
    - Ajouter . à la variable PATH (déconseillé)
      - $ PATH=$PATH:.
    - Indiquer le chemin du répertoire ou se trouve le script:
      - ./essai
  - Il faut rendre le script exécutable : $ chmod +x essai

Remarque :

le suffixe .sh ajouté à la fin du nom du script **n’a aucune importance**; il ne s’agit que d’une information pour l’utilisateur. Le système Unix ne tient aucun compte des suffixes éventuels des noms de fichiers.

TD/TP 2

1. Quel est le résultat de la commande suivante :
   - $ # echo bonjour
2. Avec une seule commande d'affichage, affichez sur deux lignes le message suivant:
   - Bonne
   - Journée
3. Écrire un script qui affiche la date en permanence, chaque écriture écrasant la précédente.
4. Écrire un script qui affiche sous forme de tableau dix étudiants, et leur classement à l'ordre inverse (en utilisant printf et des espaces).
   - Etudiant1 10
   - Etudiant2 9
   - ….
   - Etudiant10 1