UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Licence Info
Info0302
Ch Jaillet

(TP) Linux : *shell-scripts*

Info0302, "Stage Unix - *scripting*", traite des bases de Linux et des *shell-scripts*.
Ce support, qui suppose la connaissance des bases de Linux, est consacré à l'étude des shell-scripts.

support Parmi les nombreuses références qui traitent des shellscripts, nous avons retenu celui que Mathieu Nebra a mis au point pour OpenClassrooms : la partie "Automatisez vos tâches avec des scripts Bash" peut être considérée comme un cours pour cette partie d'Info0302.

- version Web https ://openclassrooms.com/fr/courses/43538-reprenez-le-controle-a-laide-de-linux/42867-introduction-aux-scripts-shell
- version pdf (pp. 367-405)
  http ://user.oc-static.com/pdf/12827-reprenez-le-controle-a-l-aide-de-linux.pdf
- licence : Creative Commons 6 2.0

A noter : (nuance) cela ne nous engage pas à vénérer VI ;-)

memento Aide mémoire des commandes Bash http ://www.misfu.com/commandes-bash.html

Ch Jaillet, département MMI, UFR Sciences, Université de Reims Champagne-Ardenne

---

Exercice 1 (Découverte : différentes façons d'exécuter une commande complexe)

- tapez la ligne de commandes
  `som=3 ; for i in 7 12 5 ;do som=som+x ;done ; echo som`
- modifiez la pour utiliser la valeur des variables `som` et `x`
  ... puis pour que le calcul s'effectue réellement (!)
- tapez tout ce texte dans un fichier `petite_somme.sh`, sur plusieurs lignes ; n'oubliez pas la ligne classique qui permet d'en faire un *script* ; ajoutez votre nom en commentaire
- exécutez ce script en le faisant lancer par *bash* : `bash petite_somme.sh`
- rendez le fichier exécutable et exécutez-le directement : `./petite_somme.sh`
- déplacez le fichier exécutable dans le répertoire `~/bin` [qui contiendra vos scripts, et que vous avez ajouté au *path*), puis lancez la commande, DIRECTEMENT : `petite_somme.sh`

Exercice 2 (Paramètres d'une commande)
1°) Ecrivez un script calculant la somme des entiers positifs au plus égaux à une valeur donnée :

- Les deux stratégies d'initialisation :
  - la valeur est lue au clavier [dans le script]
  - la valeur est passée sur la ligne de commande
- Trois possibilités pour le traitement :
  - vous pouvez utiliser une boucle TANTQUE
  - ou une boucle POUR
  - vous pouvez également utiliser un `for` en passant les valeurs en extension : commencez pas tester la commande `seq 5 12`

⇒ Réalisez les scripts correspondants (testez les différentes possibilités).

2°) [$1 + *shift*] écrivez un script calculant la somme de toutes les valeurs passées sur la ligne de commande

3°) application : écrivez un script générant un fichier HTML qui formate sous forme de liste les différents éléments de texte passés sur la ligne de commande.

[$♯] Ce script doit en premier lieu vérifier qu'on lui passe au moins deux paramètres : le nom du fichier [sans l'extension *.html*] ; un premier élément de texte.

Exercice 3 (Algorithmique ; fonctions)
1°) Ecrivez un script permettant de calculer la première puissance de deux au moins égale à une certaine valeur passée en paramètre de la commande.

2°) Modifiez le script précédent pour que la valeur soit calculée par une fonction (que vous devez écrire !) : le script ne fait alors qu'afficher une phrase donnant le résultat.

3°) Ecrivez une commande calculant le nombre de nombres premiers dans un intervalle (les deux bornes sont passées en paramètre ; elles se sont pas forcément données par ordre croissant).

4°) Construisez une fonction itérative qui calcule la factorielle dans un shellscript.

5°) Ecrivez maintenant la version récursive de cette fonction.

Exercice 4 (Travail sur des fichiers (1))
L'option `-s` de la fonction `ls` permet d'afficher en première colonne le nombre de blocs occupés par un fichier (en général, 1 bloc = 1024 octects : comparer les sorties de `ls -l` et `ls -lh`).
Ecrivez une nouvelle commande qui permet d'obtenir la place réelle du fichier sur le disque (multiple du nombre de blocs), ainsi que la place perdue. La sortie de cette commande est la suivante :

```
$ ./tailles rep5
Size   Real   Lost  Files
56148  57344  1196  tp5.ps
3420   4096   676   tp5.dvi
3130   4096   966   tp5.tex
96     1024   928   makefile
62794  66560  3766  TOTAL
```

Exercice 5 (Travail sur les fichiers (2))
1°) Ecrivez un shellscript récupérant les informations disponibles sur un fichier.

```
$ ./info tp4.tex
Nom du fichier :  tp2.tex
Repertoire :      /home/chj/enseignement/Info0302/support
Taille :          1761 octets
Proprietaire :    chj
Droits :          -rw-r--r--
Cree le :         sep 3 16:28
```

2°) Ecrivez une version de ce script acceptant des noms de fichiers génériques (du type `info *.tex`). La commande s'exécutera alors successivement sur tous les fichiers correspondants.

Problèmes d'application :

Exercice 6 (Mise en forme des compositions d'étudiants)
A l'occasion d'un TP-test de 3 heures, les étudiants ont 5 exercices à traiter.
Ils doivent créer un répertoire à leur nom et, à l'intérieur, un répertoire par exercice (dans chaque répertoire ils produisent des fichiers). Faisons l'hypothèse qu'il faut produire des fichiers texte : fichiers de type *readme*, shell-scripts, codes Python, ...
On peut imaginer qu'un des étudiants produise deux répertoires, `ex3` et `exercice1` (visibles pour vous dans cet ordre), et qu'il y ait des fichiers dans `exercice1` ainsi qu'à la base de son répertoire nominatif, mais rien dans `ex3`.
Les répertoires des étudiants sont récupérés à la fin de l'épreuve : reste à corriger !
Pour simplifier le parcours de chaque répertoire d'étudiant par l'enseignant responsable, on envisage la solution suivante... que vous devez mettre en œuvre :
- dans chaque répertoire d'étudiant vous créez un répertoire `vrac` (s'il n'existe pas)
- vous y transférez tous les fichiers qui sont à la racine du répertoire de l'étudiant (tous les fichiers ; pas les répertoires)
- à ce stade vous êtes en mesure de rendre compte de la structure globale du répertoire de l'étudiant (commande `tree`)
- ensuite, pour chaque répertoire, vous pouvez rendre compte de son contenu (commande `contenuRep`)

La commande `tree` existe ; vous devez écrire la commande `contenuRep` (*script* `contenuRep` ou `contenuRep.sh`) ; le traitement d'un répertoire d'étudiant s'appellera `traitementEt` (ou `traitementEt.sh`). Les scripts pour le traitement d'un exercice ou d'un étudiant prennent en paramètre le nom du répertoire en question.

- `contenuRep` : pour chacun des fichiers du répertoire considéré, la commande affiche le nom du fichier (`echo`) puis son contenu (`cat`)
- `traitementEt` : on affiche le contenu global du répertoire (`tree`), puis on appelle `contenuRep` pour chaque répertoire ; le texte généré est écrit dans un fichier texte (redirection de la sortie standard) ; le texte est ensuite formaté en fichier *postscript* (commande `a2ps`), puis en fichier pdf (`ps2pdf`)
- Si vous souhaitez disposer de l'ensemble des compositions des élèves en un seul fichier, il suffit d'écrire une commande `traitementTous`, qui appelle `traitementEt` pour chaque répertoire du répertoire courant : reste à concaténer les fichiers pdf générés ... avec la commande `pdftk`.

Exercice 7 (page Web locale)
On dispose sur notre compte personnel d'un certain nombre de fichiers d'image au format JPG.

1°) Ecrivez un script permettant de disposer de la liste des fichiers en question avec leur chemin absolu.

2°) Transformez le script afin que la sortie soit un fichier *html* donnant les chemins correspondants.

3°) (nouvelle version) Faîtes en sorte que ce soient les liens vers les images qui soient listés. correspondants (testez ! !)

4°) Pour chaque image rencontrée vous ajouterez dans le répertoire qui la contient une miniature de l'image (commande `convert`) et ferez en sorte que la base du lien soit cette miniature.