

TP2 : Ecriture de fichiers de commande sous Linux

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 avec 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 s'on donne les droits d'exécution au fichier et que le fichier est placé 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 l'éditeur 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.
- 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

3.1 Tester l'existence de fichier et/ou répertoire

1. Créer un fichier de commandes **testfic** qui reprend l'exemple du cours (ci-contre).
2. Affichez l'aide de la commande.
3. Testez cette commande pour un fichier et un répertoire existants/non existants.
4. Commentez le code :

```
#!/bin/bash
if test $# -eq 0 -o $1 = "-h"; 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
```

- Que représentent le `$#` et le `$1` ?
- Que fait `if test $# -eq 0 -o $1 = "-h"` ?
- Que représente `[ -f $1 ]` ?

- 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), écrire une autre solution qui n'utilise pas la commande shift.

3.2 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 **backup** 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

3.3 Date du jour

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 affiche 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 chaine "ral"

```

4 Test de commandes plus avancées

4.1 Commande umask

Tapez la commande **umask**, de manière à ce que les fichiers lors de leur création aient par défaut les droits 640 (rw-r-----), et les répertoires 750 (rwxr-x---).

Solution:

Pour un fichier masque maximum pour un fichier :	666 rw-rw-rw-
à retirer	026 ---w-rw
droit par défaut	640 rw-r-----

Pour un répertoire :

Pour un fichier masque maximum pour un fichier :	777 rwxrwxrwx
à retirer	027 ---w-rwx
droit par défaut	640 rwxr-x---

Effectuez l'opération par la commande : `umask 027`

Créez un répertoire et un fichier et vérifiez que les droits par défaut ont bien changé.

4.2 Commande grep

Créer un répertoire **essai-grep** dans votre home directory. Dans ce répertoire créer les fichiers suivants: **tomate poire pomme cerise Fraise fraise courgette POMME3 afraise**

Editez les fichiers (sortie de la commande **ls** redirigée vers **grep**) avec les critères suivants sur leur nom :

Critère 1 : Le nom doit être Fraise ou fraise

Solution : `ls | grep "^[fF]raise$"`

Critère 2 : les noms se terminent par se

Solution : `ls | grep "se$"`

Critère 3 : ai est présent dans le nom

Solution : `ls | grep "ai"`

Critère 4 Nom contenant un chiffre numérique

Solution : `ls | grep "[0-9]"`

4.3 Les filtres

Rendez-vous à la page suivante et effectuez les exercices proposés en créant les fichiers demandés et en effectuant les commandes indiquées. Il s'agit d'expérimenter par vous-même des commandes de découpage, de tri, de filtrage sur des fichiers.

<http://www.funix.org/fr/unix/filtres.htm>

Ce site web propose un cours sur UNIX très complet que vous pouvez télécharger librement.