UNIVERSITÉ Grenoble Alpes

Grenoble INP ensimag

Unix et les scripts Shell

—

Ensimag 1A

Matthieu Moy, Grégory Mounié

2018-2019

1 Les scripts shell

Les commandes que vous tapez dans un terminal sont exécutées par un shell. Par défaut, sous Linux, ce shell est `bash`. Ces commandes peuvent être mises dans un fichier qui sera exécuté dans son ensemble par le shell, le script. Ce TP vous propose de construire des petits morceaux de scripts. Il s'appuie sur un poly en PDF, sur le Bourne Shell, disponible sur la page du cours.

1.1 Deux variantes pour ceux maitrisant déjà `bash`

1. Vous pouvez faire ce TP avec un autre shell que vous voulez apprendre, comme `Zsh`.
2. Vous pouvez vous grouper en binôme pour réaliser une galerie d'images WEB en shell : cela consiste à générer une page web (HTML) à partir d'un répertoire contenant des images. Vous pouvez générer aussi des vignettes (petites images) pour la page de garde afin d'économiser de la bande passante et accélérer son chargement. Vos vignettes pointeront ensuite vers les images en grand format. Un sujet avec des indications est disponible sur la page du cours.

2 Utilisation de la ligne de commande

En guise d'échauffement, nous allons voir quelques petites astuces qui rendent la vie plus pratique sous Unix. Commencez par créer un fichier `toto.txt` dans un répertoire de votre choix (par exemple, `~/tp-unix/`). Par exemple, `touch toto.txt` dans le bon répertoire.

Maintenant, nous changeons (déjà !) d'avis, et souhaitons ranger ce fichier dans un sous-répertoire `travail`, qui n'existe pas encore.

Question 1 *Créez le répertoire, avec* `mkdir travail`*.*

Maintenant, on veut rentrer dans ce répertoire, mais on ne veut pas re-taper le mot `travail` en entier (oui, un bon informaticien est un informaticien flémard !). Deux solutions : reprendre la ligne ci-dessus et l'éditer, ou bien en taper une nouvelle en utilisant une formule magique :

Question 2 *Appuyez sur la touche « Flèche haut » pour retrouver la commande précédente, puis « Control-a » pour revenir en début de ligne, puis « Alt-d » (ou « ESC d » si ça ne marche pas) pour effacer le premier mot. On peut maintenant taper « cd » et obtenir la ligne souhaitée. Mais comme on est curieux, on ne va pas faire comme ça, on fait « Control-a » puis « Control-k » pour effacer la ligne, et on passe à la question suivante.*

Ceci est valide avec la configuration `set -o emacs` de `readline`. Vous diviserez par deux le nombre de *keystrokes* en adoptant le mode vi (`set -o vi`) !

Question 3 *Taper* `cd !$` *puis Entrée. Le shell va remplacer le* `!$` *par le dernier argument de la dernière ligne, et va en fait exécuter* `cd travail`*.*

Notez que si vous avez les doigts gourds et que vous tapez `cd !*`, car `*` se trouve juste sous `$`, alors il se passera, coup de chance, la même chose ! En effet, `!*` représente toute la ligne sauf le premier argument, ce qui est identique pour le cas présent.

Nous y voilà, on va maintenant déplacer le fichier, qui est maintenant `../toto.txt` dans le répertoire courant. Bien sûr, hors de question de taper le nom du fichier en entier :

Question 4 *Taper* `mv ../to`*, puis appuyez sur TAB. Le shell complète tout seul. D'une part, ça vous économise quelques caractères, mais en plus, ça vous évite de faire des fautes de frappe ! Compléter la ligne avec* `.` *pour définir comme cible le répertoire courant.*

Bon, le fichier est là, mais on change encore une fois d'avis. Finalement, ce n'est pas un fichier `.txt` qu'on veut, mais un `.tex`. Bien sûr, si on était fort, on taperait directement `mv toto.txt !#:1:s/.txt/.tex/`, mais finalement, c'est un peu compliqué, on va faire autrement :

Question 5 *Tapez* `mv toto.{txt,tex}`*. Le shell va remplacer l'argument avec des accolades par plusieurs arguments :* `toto.` *suivi de la première solution,* `txt`*, puis la seconde (séparée par un espace), donc au final, la commande exécutée est* `mv toto.txt toto.tex`*.*

Voilà, l'échauffement est terminé, en espérant que ça vous a donné envie d'en apprendre plus sur votre shell favori (avez-vous déjà tapé `man bash` ?), voire d'explorer les possibilités d'autres shells plus puissants, comme `zsh`.

3 Premiers scripts

3.1 Des nombres ...

Poly : pages 10 (paramètres), 44 (if) et 59 (test).

Question 6 *Ecrire un script shell qui lit 2 nombres entrés en arguments par l'utilisateur et qui affiche une phrase, bien évidemment correcte, du type* `Le nombre 3 est inférieur au nombre 5` *(dans le cas où les deux nombres entrés sont* 3 *et* 5*).*
Attention, penser à gérer le cas de l'égalité.

indice : La commande `test` *XXX* `-le YYY` teste si *XXX* est plus petit ou égal à `YYY`. La commande `test` *XXX* `-eq YYY` teste l'égalité. À utiliser avec un if/then/elif/else approprié. Attention, les espaces comptent, il en faut autour de l'opérateur `-eq` (qui est un argument donné à la commande `test`)

3.2 Quelques manipulations de fichiers

Poly : pages 10 (paramètres), 43 (for) et 60 (test -d).

Question 7 *Ecrire un script shell qui liste tous les dossiers du répertoire courant (sans lister les sous-répertoires).*

indice : Une solution : parcourir tous les fichiers et répertoires avec une boucle `for`, et à chaque tour de boucle vérifier si on a bien un répertoire avec `test -d`

Question 8 *Modifier le script précédent afin de préciser en paramètre le chemin du répertoire que l'on souhaite lister (et non plus le répertoire courant).*

indice : On peut utiliser la commande `cd` pour se placer dans le bon répertoire. Les arguments du programme sont `"$1"`, `"$2"`, ...

Question 9 *Modifier à nouveau le script précédent afin de renvoyer un message d'erreur si le chemin passé en paramètre n'est pas un répertoire valide.*

indice : Tester avec `test -d` si le répertoire est valide.

Question 10 *Modifier une dernière fois le script précédent afin de pouvoir lister plusieurs chemins passés en paramètres.*

indice : Utiliser une boucle `for`. L'ensemble des paramètres passé au script est accessible via `"$@"`.

3.3 Manipulations de fichiers avancées

Poly : pages 5 (redirections) et 43 (for).

Question 11 Exécuter le script `tp_pas_a_pas.sh` fourni (sur la page web du cours). Celui-ci crée 5 répertoires (dir1, ..., dir5) contenant chacun 5 fichiers (file1.txt, ..., file5.txt). Utiliser le script précédent afin de vérifier que la création des 5 répertoires a bien eu lieu. *Ecrire un script shell déplaçant chaque fichier contenu dans dir1 à dir5 dans un dossier* `ALL` *en* renommant *chaque fichier de façon à ajouter au début de chaque nom de fichier le nom du répertoire d'origine. Penser à supprimer les dossiers d'origine.*
En cas d'erreur, réexécuter le script fourni afin de recréer les dossiers ainsi que leur contenu.

indice : Une boucle `for d in dir*; do ...` vous permettra d'effectuer une action pour chaque répertoire. Il faudra une deuxième boucle (imbriquée) pour parcourir les fichiers du répertoire `$d`.

Question 12 *Écrire un script shell qui ajoute à l'intérieur de chaque fichier du répertoire ALL un commentaire indiquant sa* date *de dernière modification.*

indice : Pensez à utiliser la redirection » pour ajouter quelque chose à la fin du fichier en conservant son contenu initial.

3.4 Commandes Unix classiques

Poly : page 5 (redirections).

Question 13 *Ecrire un script shell qui, à partir de la commande* `last`*, affiche les logins des dernières personnes s'étant connecté sur votre machine, avec le nombre de connexions par utilisateur. Si possible, triez cette liste par nombre de connexions. Si la sortie de* `last` *n'est pas intéressante sur votre machine, faites la même manipulation sur* `pcserveur.ensimag.fr` *qui est utilisée par plus de monde.*

indice : Pour supprimer les deux dernières lignes de la sortie d'une commande, on peut utiliser `head -n -2`. L'option `-c` de la commande `uniq` permet d'afficher le nombre de répétitions successives d'une ligne (à utiliser en conjonction avec `sort`). La commande `sort -n` permet de trier le tout.

Question 14 *Écrire un script shell qui renomme, dans le répertoire courant, tous les fichiers comportant des blancs dans leur nom, en supprimant les blancs dans le nouveau nom. Si le temps le permet, faites en sorte que si un fichier avec le nouveau nom existe déjà, un numéro soit ajouté au nom pour le rendre unique.*

indice : Étant donné une chaîne `$f` contenant des espaces, `echo "$f" | sed 's/ //g'` permet de supprimer ces espaces [1].

1. En fait, on devrait plutôt utiliser `printf '%s' "$f"`, plus robuste et plus portable que `echo "$f"`

3.5 xkcd

Poly : pages 5 (redirections) et 43 (for).

On se propose d'écrire un petit script d'une ligne pour télécharger des webcomics, en l'occurrence xkcd (`http://xkcd.com`). Un aspirateur de site web n'est en général pas suffisant car les images sont triées localement par ordre alphabétique et non par ordre d'apparition sur le site.

Attention : l'auteur de xkcd serait probablement flatté de savoir qu'il est cité dans un TP d'Ensimag, mais n'appréciera sans doute pas si nous abusons de sa bande passante. Merci donc de rester courtois dans vos essais, en essayant vos lignes de commandes sur *quelques* images (i.e. ne pas faire des dizaines de requêtes à chaque tentative, et surtout, ne pas laisser tourner un script en boucle).

Une petite analyse du site permet de voir que la $i^{\text{ème}}$ image se trouve sur la page web `http://xkcd.com/i`. Par exemple, la page web de la $3^{\text{ème}}$ image est `http://xkcd.com/3`.

Question 15 *Écrire une boucle* for *permettant d'afficher toutes les urls correspondantes aux images de 1 à 15. On utilisera pour ce faire les commandes* seq [2] *et* echo.

On cherche maintenant à récupérer sur la page web l'URL de l'image qui nous intéresse. Dans un premier temps, oublions la boucle précédente, et ne travaillons que sur une seule page. Une analyse du code html nous permet de voir que le code de chaque page est très semblable. En particulier, l'URL se trouve sur une ligne contenant le mot "hotlinking". Par exemple, pour la première image :

```
<h3>Image URL (for hotlinking/embedding): http://imgs.xkcd.com/comics/barrel_cropped_(1).jpg</h3>
```

On se propose de télécharger la page web et de l'afficher sur la sortie standard. Pour se faire, on peut utiliser par exemple "`wget -O -`" (attention, O majuscule ici) ou "`curl -o -`". Il reste alors à extraire la ligne qui nous intéresse à l'aide de la commande *grep*. Enfin, dernière étape, extraire de cette ligne l'URL. On utilise une fois de plus *grep*. La commande extrayant l'URL est donc au final :

```
wget http://xkcd.com/1 -O - | grep hotlink | grep -o 'http.*jpg'
```

Assurez-vous d'avoir bien compris cette ligne avant de continuer.

Il est donc désormais possible d'itérer sur chaque page, d'extraire à chaque fois l'URL de l'image, et de télécharger l'image à l'aide de *wget*. Néanmoins le problème de l'ordre alphabétique reste encore à régler. Pour classer les images dans le bon ordre, il suffit de les renommer par un numéro. La commande *printf "%03d" 5* remplace le chiffre "5" par "005". Cette commande permet ainsi de s'assurer que "5" et "10" seront ordonnés correctement par ordre alphabétique (5 arrive après 10 mais 005 arrive avant 010). *wget -O nom_de_fichier* permet de plus de stocker directement l'image à télécharger avec le bon nom de fichier.

Question 16 *Écrire le script complet (qui peut tenir en une seule ligne) permettant de télécharger et de renommer les images pour pouvoir les lire dans l'ordre, localement.*

2. Non-POSIX mais facile à ré-implémenter si besoin