ESIR-SYS1 TP 2-3-4: Réaliser un Script Shell (TP Non Noté, 0.5 point bonus possible)

François Taiani (`ftaiani.ouvaton.org`)*

1 Préliminaires

1.1 Enregistrement de votre binôme

Commencez par enregistrer votre binôme sur Moodle, dans le groupe de TP auquel vous appartenez. Les binômes contiennent deux étudiants. Si votre groupe de TP contient un nombre impair d'étudiants vous pouvez exceptionnellement travailler à 3 **après autorisation de votre enseignant de TP**.

1.2 Préambule

- La première ligne d'un script doit être : `#!/bin/bash` ;
- il faut donner les droits d'exécution au fichier script (`chmod`) ;
- en phase de mise au point, on peut tracer ce que fait le shell en exécutant le script avec la commande : `bash -x script arguments` ;
- vous êtes libre d'utiliser l'éditeur de votre choix pour ce TP (*Visual Studio Code*, `nano`, `emacs`, `gedit`), privilégiez cependant un éditeur capable de coloration syntaxique (*Visual Studio Code*, `emacs`, `gedit`), en utilisant une police à chasse fixe (*fixe-width font*) comme `courrier`, qui rendent la programmation plus aisée.

1.3 But du TP

Le but de cet exercice est de réaliser un script qui produit un album photo pdf de format carré à partir de photos de différentes tailles.

1.4 Commandes utiles

Nous allons utiliser les commandes suivantes de manipulation d'images et de fichiers PDF :

- `convert` (de la suite `https://imagemagick.org`) permet de transformer puis convertir des images entre plusieurs formats. Nous allons en particulier utiliser la syntaxe suivante :

  ```
  convert -gravity center -crop WIDTHxHEIGHT+0+0! input.jpg output.jpg
  ```

  qui permet de rogner (`crop` en anglais) l'image stockée dans le fichier `input.jpg` aux dimensions `WIDTH` × `HEIGHT` (exprimées en pixels) puis de la sauvegarder dans le fichier `output.jpg`. (L'option `-gravity center` et l'indication de décalage `+0+0!` permettent de centrer la fenêtre de rognage.)
- `identify` va nous permettre d'extraire la dimension la plus petite des images que nous allons traiter (pour la plupart rectangulaires). Nous allons utiliser la syntaxe qui suit :

  ```
  identify -format "%[fx:min(w,h)]" input.jpg
  ```

  qui imprime sur la sortie standard la dimension en pixels (`w`*idth* ou `h`*eight*) la plus petite (`min`) de l'image contenue dans le fichier `input.jpg`.

*francois.taiani@irisa.fr

- `img2pdf` permet de convertir les images `jpg` en fichiers PDF, en utilisant la syntaxe suivante

  ```
  img2pdf --output output.pdf --pagesize 10cmx10cm input.jpg
  ```

  qui transforme le fichier `input.jpg` dans le fichier `output.pdf`, sur une taille de papier spécifiée par l'option `--pagesize` (ici 10cm×10cm).

1.5 Substitution de commandes

Pour rogner chaque image en carré, il va être nécessaire d'extraire la dimension la plus petite de l'image (avec `identify`) puis d'utiliser cette dimension pour renseigner les arguments de l'option `-crop` de la commande `convert`. Pour faire cela vous aurez besoin de l'opérateur de substitution `$(..)` (aussi noté de façon équivalente `` `..` ``) qui permet d'insérer la sortie d'une commande à l'intérieur d'une autre ligne de commande. La ligne suivante par exemple

```
my_var=$(uname)
```

exécute la commande `uname`, remplace `$(uname)` par la sortie de `uname` (`"Linux"` dans ce cas), dans la ligne de commande qui devient donc `my_var=Linux`, puis l'exécute, assignant la chaîne `"Linux"` à la variable de Shell `my_var`.

1.6 Jeu d'images

Nous allons travailler sur un jeu d'images de la photothèque de l'université disponible à l'adresse `https://foad.univ-rennes1.fr/pluginfile.php/1692008/mod_resource/content/0/imagesUR1.zip`. Ce fichier est aussi disponible sur la page Moodle du cours (`https://foad.univ-rennes1.fr/course/view.php?id=1000428`).

Une fois téléchargée, pensez à décompresser l'archive (avec `unzip`) dans un répertoire créé dans ce but (en utilisant `mkdir` pour créer le répertoire, et ensuite l'option `-d` de `unzip`). Par exemple :

```
mkdir imagesUR1
man unzip
unzip imagesUR1.zip -d imagesUR1
```

2 Travail demandé

2.1 Partie I : Version initiale (script `sys1_part_1.sh`)

Réalisez un script `sys1_part_1.sh` qui prend en argument de ligne de commande le nom d'un fichier contenant une image jpg, puis réalise le traitement suivant. Le script

- obtient la dimension la plus petite de l'image et la stocke dans une variable de shell ;
- calcule le nom d'un fichier de sortie intermédiaire en replaçant la terminaison `".jpg"` du fichier d'entrée par la terminaison `"_square.jpg"`, et stocke ce nouveau nom dans une variable. Pour faire cela, utilisez la substitution de motif de bash (*pattern substitution*). Pour découvrir ce mécanisme, tapez `man bash`, puis cherchez (en tapant sur `/`) la chaîne "*trailing portion*" pour vous documenter sur la syntaxe `${parameter%word}`. Quittez `man` en tapant `q`.
- de même, calcule le nom du fichier final de sortie en replaçant la terminaison `".jpg"` du fichier d'entrée par la terminaison `".pdf"`, et stocke ce nouveau nom dans une variable.
- utilise `convert` pour rogner l'image en carré, et la sauvegarder au format jpg dans le fichier intermédiaire.
- utilise `img2pdf` pour produire un document PDF à partir du fichier intermédiaire, et le sauvegarder dans le fichier final.

Testez votre script avec plusieurs types de chemins de fichier (dans le même répertoire, absolu, relatif) et plusieurs images.

Quelques conseils:

- Évitez d'utiliser directement les variables automatiques `$1`, `$2`, etc., mais assignez leur contenu à des variables de shell avec des noms explicites en début de script. Cela facilitera la compréhension de votre code.
- Si une variable peut contenir des espaces (c'est le cas des noms de fichier par exemple), il est conseillé de les mettre entre guillemets doubles (par exemple `ls "$nom_de_fichier"` plutôt que `ls $nom_de_fichier`) quand vous les utiliser, pour éviter une mauvaise interprétation des espaces et augmenter la portabilité de votre code. `bash` (le shell de Linux) n'a en général pas besoin de guillemets, mais d'autres shell (comme `sh`) si.
- Lorsque le contenu d'une variable est concaténé à d'autres caractères, la syntaxe `${SOME_VAR}` peut être utile pour éviter que les caractères concaténés ne soient confondus avec le nom de la variable, comme par exemple dans `${SOME_VAR}x${SOME_VAR}`.
- Insérer quelques `echo` dans votre code pour informer l'utilisateur du progrès de votre script, et faciliter la mise au point de votre code.
- Pensez à commenter votre code afin de le rendre intelligible. Un bon commentaire ne paraphrase pas le code mais en explique l'intention, et clarifie son fonctionnement.

2.2 Partie II : Ajouter des vérifications (script `sys1_part_2.sh`)

Même fonctionnement que la version précédente, mais le script doit faire les vérifications suivantes et émettre un message puis se terminer en cas d'erreur :

- nombre d'arguments correct (1) ;
- existence du fichier passé en argument ;
- non-existence du fichier intermédiaire ;
- non-existence du fichier PDF final.

Pensez à tester ensuite tous les cas d'erreur pour vérifier que votre script fonctionne comme prévu.

2.3 Partie III : Traiter une liste de fichiers (script `sys1_part_3.sh`)

On veut avoir la possibilité d'appeler le script avec un nombre de fichiers quelconque afin d'effectuer le traitement ci-dessus sur chacun d'eux.

Commencez par mettre tout le code du script précédent dans une fonction.

La syntaxe de définition d'une fonction est :

```
# parameters :
# first value to be handle
# second value to be handled
# ...
function myFonction
{
  # retrieving arguments
  local var1=$1
  local var2=$2
  #...
}
```

- Une fonction peut contenir toute commande ou instruction valide dans un script.
- Pour sortir d'une fonction avant la fin sans sortir du programme, on peut utiliser l'instruction `return`.
- La syntaxe d'appel d'une fonction est : `maFonction $variable1 $variable2`
- Par défaut les variables utilisées dans une fonctions sont de portée **globales** (!). Le mot clé `local` permet de limiter la visibilité d'une variable à la fonction, et de protéger les variables éventuelles de même nom qui existerait ailleurs dans le script.

Remarque : une fonction doit etre définie avant toute instruction d'appel.

- Programmez une itération qui appelle cette fonction avec chaque argument du script en paramètre.
- Testez
  - avec un seul argument
  - avec plusieurs arguments
  - avec une liste vide

  en n'oubliant pas les différents cas du script précédent.

2.4 Partie IV : Générer l'album et nettoyer (script `sys1_part_4.sh`)

Ajoutez maintenant au script les fonctionnalités suivantes :

- Le script utilise maintenant un *répertoire de travail temporaire*. Ce répertoire de travail (dont vous choisirez le nom) doit être créé en début de script, et détruit à la fin. Pensez à vérifier que ce répertoire n'existe pas déjà, et à arrêter le script avec un message d'erreur s'il existe déjà.

  Attention, pour détruire ce répertoire vous utiliserez les commandes `rm` et/ou `rmdir`. Mal utilisées, ces commandes sont *dangereuses* (en particulier combinées avec le caractère d'expansion `*`). Prenez garde à limiter leur portée pour ne pas détruire malencontreusement vos fichiers.
- Le script traite chaque fichier `jpg` passé en paramètre comme précédemment, mais sauve maintenant les fichiers `_square.jpg` et `pdf` résultants dans le répertoire de travail temporaire.
- Chaque fichier `pdf` produit doit être redimensionné à la taille 10cm × 10cm en utilisant l'option `--pagesize` de `img2pdf` (voir plus haut).
- Enfin, quand tous les fichiers `pdf` ont été produits et redimensionnés, le script utilise la commande `pdfunite` pour concaténer tous ces fichiers en un seul album (dont vous choisirez le nom) sauvegardé dans le répertoire de travail du script. Utilisez `man pdfunite` pour vous renseigner sur la syntaxe de cette commande. Utilisez l'expansion de noms de fichiers pour réaliser ce traitement. Pensez à vérifier que le fichier album n'existe pas déjà et à interrompre le script avec un message d'erreur si c'est le cas.

3 Soumission et attendus pour obtenir 0,5 point bonus

Vous êtes fortement encourager à soumettre votre TP pour obtenir 0,5 point bonus sur votre note finale de TP.

3.1 Soumission

- Enregistrez votre binôme sur le site Moodle du cours.
- Emballez l'ensemble de vos scripts (`sys1_part_?.sh`) dans une archive `.zip` (par exemple avec la commande `zip`), et soumettez votre archive sur le site Moodle du cours. Attention à respecter la date de soumission indiquée sur le site Moodle du cours. Les rendus en retard ne seront pas comptabilisés.

3.2 Attendus

Pour obtenir 0,5 point bonus, votre TP, sans être parfait, doit marcher convenablement.

4 Collaboration et entraide

Le TP ne donnant lieu qu'à 0,5 point bonus vous êtes fortement encouragé(e)s à collaborer avec vos camarades pour comprendre et progresser sur ce TP.

Attention cependant à ne pas simplement copier/coller du code que vous ne comprendriez pas : une telle approche ne vous permettra pas d'acquérir les compétences nécessaires à votre réussite lors du contrôle de TP sur table. À la fin du TP, vous devez être capables de reproduire une solution fonctionnelle au problème qui a été posé.

— Fin du sujet de TP —