TP

Langages de programmation script

Script Shell : bash

Exercice 1 : familiarisation

Faire un script (exo1) en bash qui affiche "Bonjour Monde" à l'écran

Résultat:

$sh ./exo1

coucou

Conseil : se servir de la commande echo

Exercice 2 : familiarisation

Faire un script exo2 qui prend en argument un nom de fichier, affiche "x sera sauvegardé en x.save", et effectue la copie de x dans un fichier nommé x suivi de .save

le contenu du répertoire avant:

$ls

fichier1

j'exécute mon script :

$sh ./exo2 fichier1

fichier1 sera sauvé en fichier1.save

le contenu du répertoire après :

$ls

fichier1 fichier1.save

Exercice 3 : introduction à for

Construire un script qui prend en argument le nom d'un dossier (qu'on appelle $SOURCE_DIR).

il créé un un dossier (qu'on appelle $DEST_DIR) qui se nomme $SOURCES_DIR concaténé à .save

il copie l'intégralité du dossier $SOURCES_DIR dans $DEST_DIR, en ajoutant '.save' à la fin de chacun des fichiers ainsi copiés.

le script doit annoncer chacune de ces actions par une phrase à l'écran (uniquement de l'affichage)

exemple :

le dossier 'photos' contient 'photo1.jpg' et 'photos2.jpg'. l'execution du script donne :

$ sh ./exo3.sh photos

création du repertoire photos.save

copie de photos/photo1.jpg dans photos_save/photo1.jpg.save

copie de photos/photo2.jpg dans photos_save/photo2.jpg.save

et on doit trouver un dossier nommé 'photos.save' qui contient 'photo1.jpg.save' et 'photo2.jpg.save'

Exercice 3 : introduction à for

Conseil : utiliser for et les "`" pour faire la boucle. par exemple:

```
for i in `ls`

do

  echo ${i}

done
```

se rappeller que la concaténation en bash est automatique. ex :

```
VAR_A="bon"

VAR_B="jour"

VAR_C=${VAR_A}${VAR_B}

echo ${VAR_C}
```

Remarque1 : les {} dans l'appel d'une variable servent à enlever l'ambiguité. $VAR est identique à ${VAR}. par contre VAR_1VAR_2 est ambigüe (si $VAR_2="toto", alors VAR_1VAR_2 est équivalent à $VAR_1toto ???) alors que ni ${VAR_1$VAR_2} ni ${VAR_1}${VAR_2} ne le sont...

Remarque2 : les ` servent à exprimer le résultat d'un commande. par exemple pour mettre la liste des fichier contenus dans un repertoire dans une variable on peut faire VAR_1=`ls`

Exercice 4 : pipes et constructions de fichier.

Le but de cet exercice est de faire un script qui va créer une page html contenant des liens vers les fichiers d'un répertoire donné : une page d'index en quelque sorte.

Le script doit prendre en paramètre le nom du dossier à traiter. Il doit créer dans ce dossier un fichier "index.html". Ce fichier "index.html" doit avoir une syntaxe html valide.

Le corps de page doit en outre contenir des liens vers les fichiers contenus dans le repertoire à raison de un lien par ligne. Enfin, à la fin de son exécution, le script doit signaler qu'il a créé un fichier index.html dans le répertoire en question.

Ex :

on dispose d'un répertoire photos contenant photo1.jpg et photo2.jpg

on appelle le script pour traiter ce répertoire:

```
dyn-ecim13:~/exo lorinc$ sh ./exo4.sh photos

photos/index.html écrit !
```

le repertoire photos contient maintenant un fichier "index.html" :

```
dyn-ecim13:~/exo lorinc$ ls photos

index.html      photo1.jpg      photo2.jpg
```

Exercice 4 : pipes et constructions de fichier.

et le contenu de ce fichier index.html est le suivant :

```
<HTML>

<HEADER></HEADER>

<BODY>

<a href="photos/photo1.jpg">photo1.jpg</a><br />

<a href="photos/photo2.jpg">photo2.jpg</a><br />

</BODY>

</HTML>
```

Conseils : utiliser la redirection de flux ">". par exemple :

```
echo "coucou" > fichier_coucou
```

fait en sorte que "coucou" ne soit pas écrit à l'écran mais plutôt dans le fichier "fichier_coucou"

Attention : ">" écrase le fichier de destination s'il existe. si on veut écrire à la fin du fichier de destination, il faut utiliser ">>".

```
echo "coucou" > exempl

echo "coucou numero 2" >> exempl

cat exempl
```