UE : Systèmes d'exploitation

Licence 3 SFA Informatique

TP : Bash scripting

Ce TP se déroule individuellement. Vous devrez, d'ici la fin de la séance (3h), proposer un rapport de TP comprenant : les commandes shell utilisées, les scripts shells et les réponses aux différentes questions du sujet.

Vous mettrez la liste de vos scripts dans une archive au format .zip, .tar ou .tar.gz à déposer dans la zone Travaux sur l'ENT en fin de session.

Introduction : Les bases

Un script bash est un fichier texte qui a vocation à être **exécuté** par **bash**.

La première ligne contient le 'she-bang', il définit que le script doit **par défaut** être exécuté par un "Bourne shell compatible": **#!/bin/bash**
*Pour information et en guise d'exemple, si vous écrivez **#!/usr/bin/env python** cela a pour effet qu'il sera par défaut exécuté par python, mais ce n'est pas le but ici.*

L'utilisateur qui exécute le script doit disposer des droits d'exécution sur le fichier.

Un **script bash** a souvent une extension de fichier **.sh** plus par convention plus que par obligation.

En supposant que le script est dans votre dossier courant, il peut être exécuté comme ceci :
./script.sh arg1 arg2 arg3…
Des exemples :
./some_script.sh
./myscript.sh "coucou" ./randomfile.txt
Ici "some_script.sh" est un simple script sans argument.
Tandis que "myscript.sh" est un script qui est exécuté avec 2 arguments, le premier étant le texte "coucou" et le deuxième étant un chemin vers un fichier nommé "randomfile.txt".

Les pages suivantes donnent de **nombreuses informations utiles** au sujet des variables, boucles, conditions, etc.. N'hésitez pas à en abuser.
https://devhints.io/bash
https://github.com/LeCoupa/awesome-cheatsheets/blob/master/languages/bash.sh

Introduction : Des exemples de script

Rien de mieux que des exemples pour voir à quoi cela ressemble concrètement.

Exemple de script "hello world" :

```
#!/bin/bash
echo "Hello World"
```

Exemple de script avec des variables :

```
#!/bin/bash

greeting="Welcome"
user=$(whoami)
day=$(date +%A)

echo "$greeting back $user! Today is $day, which is the best day
of the entire week!"
echo "Your Bash shell version is: $BASH_VERSION. Enjoy!"
```

Ces exemples sont tirés de la page suivante qui référence de nombreux autres exemples plus ou moins complexes qui peuvent vous être utiles. N'hésitez pas à jeter un coup d'œil.
https://linuxconfig.org/bash-scripting-tutorial-for-beginners

Exercice 1

Créez le script "somme.sh" dans lequel seront définies 2 variables **a** et **b** et afficher la somme de ces 2 variables :

- **a** qui vaut **12**
- **b** qui vaut **30**

Exercice 2

Créez le script "bonjour_ask.sh" qui :

- Affiche à l'utilisateur "Quel est ton nom ?"
- Attend que l'utilisateur écrive son nom dans le terminal pour le stocker dans une variable
- Affiche ensuite "Bonjour" suivi du nom qui a été saisi

Exemple :

Exercice 3

Créez le script “bonjour_param.sh” qui fait la même chose que dans **l’Exercice 2**, mais au lieu de demander à l’utilisateur d’écrire son nom, faites le passer en argument du script.

Exemple :

```
damien@dgn:~/Desktop/cours/TP$ ./bonjour_param.sh Damien
Bonjour Damien
damien@dgn:~/Desktop/cours/TP$ ./bonjour_param.sh Sylvestre
Bonjour Sylvestre
```

Exercice 4

Créez le script “somme_param.sh” qui affiche la somme de 2 nombres passés en paramètres du script.

Exemple :

```
damien@dgn:~/Desktop/cours/TP$ ./somme_param.sh 10 11
21
damien@dgn:~/Desktop/cours/TP$ ./somme_param.sh 4 5
9
```

Exercice 5

Le script de **l’Exercice 4** fonctionne, mais il plante s’il manque un argument.

Modifiez le script de **l’Exercice 4** pour que celui-ci affiche l’erreur “Nombre d’arguments incorrect” si il n’y a pas exactement 2 paramètres qui ont été passés au script.
Vous appellerez ce nouveau script “somme_param_safe.sh”.

Exemple :

```
damien@dgn:~/Desktop/cours/TP$ ./somme_param_safe.sh
Nombre d'arguments incorrect
damien@dgn:~/Desktop/cours/TP$ ./somme_param_safe.sh 12
Nombre d'arguments incorrect
damien@dgn:~/Desktop/cours/TP$ ./somme_param_safe.sh 12 30
42
```

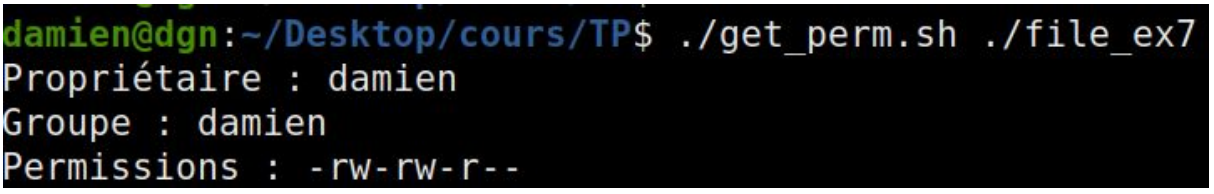
Exercice 6

Créez le script “for.sh” qui affiche les nombres 0 à 4 sur une nouvelle ligne à l’aide d’une boucle **for**.

Exemple :

Exercice 7

Créez un script “get_perm.sh” qui prend en paramètre le chemin d’accès vers un fichier et qui affiche en sortie le propriétaire, le groupe et les permissions qui s’appliquent sur ce fichier.
Les informations doivent être visuellement présentés comme dans l’exemple suivant :



Aide
Une façon de faire peut être d’utiliser les commandes **ls -l** et **cut**, mais il y a aussi d’autres solutions, à vous de voir.

Exercice 8

Prenez l'archive "TP1_ex8" présente sur l'ENT et extrayez son contenu dans un dossier.
Ce dossier contient une liste de fichiers "texte".
Chaque fichier contient le texte **SUCCESS** ou le texte **ERROR**.

Créez un script "get_success.sh", qui prend pour arguments une liste de chemins de fichiers.
Faites en sorte qu'il stocke dans un nouveau fichier "output.txt" la liste des fichiers qui contiennent le texte **SUCCESS** parmi les fichiers qui sont présents dans le dossier TP1_ex8.

Exemple :

```
damien@dgn:~/Desktop/cours/TP$ ./get_success.sh ./TP1_ex8/*
damien@dgn:~/Desktop/cours/TP$ cat output.txt
./TP1_ex8/a123.txt
./TP1_ex8/c789.txt
./TP1_ex8/g512.txt
./TP1_ex8/h152.txt
```

Aide
Pour donner plusieurs chemins de fichiers en arguments d'un seul coup à une commande ou à un script, utilisez le wildcard ***** comme dans l'exemple ci-dessus.

Ici **./TP1_ex8/*** équivaut à écrire **un par un chaque chemin de fichier** présent dans le dossier **TP1_ex8**.

En gros, c'est équivalent à ceci :

```
damien@dgn:~/Desktop/cours/TP$ ./get_success.sh ./TP1_ex8/a123.txt ./TP1_ex8/b456.txt ./TP1_ex8/
c789.txt ./TP1_ex8/d565.txt ./TP1_ex8/e124.txt ./TP1_ex8/f541.txt ./TP1_ex8/g512.txt ./TP1_ex8/h
152.txt ./TP1_ex8/i874.txt
damien@dgn:~/Desktop/cours/TP$ cat output.txt
./TP1_ex8/a123.txt
./TP1_ex8/c789.txt
./TP1_ex8/g512.txt
./TP1_ex8/h152.txt
```

C'est quand même mieux avec le wildcard ***** !

Exercice 9

Prenez le fichier “TP1_ex9.env” sur l’ENT.
Ce fichier représente une liste de configurations qui pourraient correspondre à celles d’une base de données. Observez à quoi ressemble son contenu. Le but sera de définir les configurations qui ne sont pas encore configurées (celles dont les valeurs sont entourées par des ## ##).

Créez un script “set_config.sh” qui prend 4 arguments :

- Le fichier de configuration ciblé (à configurer)
- Le nom de la BDD
- L’host de la BDD
- Le mot de passe de la BDD

Ce script devra définir dans le fichier cible les valeurs qui ont été saisies en arguments du script. Vous devrez aussi faire en sorte qu’une aide sur l’utilisation de la commande s’affiche si le nombre d’arguments qui ont été saisis n’est pas correct.

Exemple :

```
damien@dgn:~/Desktop/cours/TP$ cat TP1_ex9.env
# My super env file !

DB_USER=root
DB_NAME=##DB_NAME##
DB_HOST=##DB_HOST##
DB_PASSWORD=##DB_PASSWORD##
damien@dgn:~/Desktop/cours/TP$ ./set_config.sh
Utilisation de la commande :
./set_config.sh [fichier ciblé] [DB_NAME] [DB_HOST] [DB_PASSWORD]
Exemple :
./set_config.sh ./TP1_ex9.env mydb 127.0.0.1 azertyuiop
damien@dgn:~/Desktop/cours/TP$ ./set_config.sh ./TP1_ex9.env mydb localhost azertyuiop
damien@dgn:~/Desktop/cours/TP$ cat TP1_ex9.env
# My super env file !

DB_USER=root
DB_NAME=mydb
DB_HOST=localhost
DB_PASSWORD=azertyuiop
```

Aide
Il est possible d’utiliser la commande **sed**, mais il y a aussi d’autres moyens, à vous de voir.

Exercice 10

Prenez l'archive "TP1_ex10" présente sur l'ENT et extrayez son contenu dans un dossier.

Afin de tester des cartes électroniques externes, nous utilisons un logiciel qui crée un fichier de log pour chaque dispositif testé.
Ces fichiers de log sont nommés de cette manière : teraterm_[DATE ET HEURE].log
Exemple : teraterm_2020-09-09 10-56-10.log

À l'intérieur de chaque fichier, on retrouvera l'identifiant de la carte, ici 864376040797622 :

```
[2020-09-09 10:56:22.974] SIM800 IMEI is:
[2020-09-09 10:56:23.076] 864376040797622
```

Si le test s'est avéré réussi, on retrouvera **TEST SUCCEEDED** vers la fin du fichier :

```
[2020-09-09 11:04:50.824]
[2020-09-09 11:04:50.928] TEST SUCCEEDED
[2020-09-09 11:04:50.928]
```

Si le test s'est avéré réussi, on retrouvera aussi le **RSSI** (force du signal) correspondant :

```
[2020-09-09 11:04:15.898]
[2020-09-09 11:04:21.104] GSM SIGNAL IS GOOD AND RSSI IS: 8,
[2020-09-09 11:04:21.207]
```

Afin d'obtenir un récapitulatif des différents tests effectués, créez un script "merge_tests_as_csv.sh" qui a pour but de créer un fichier CSV (séparé par des points virgule) qui contient les informations suivantes :

- Date et heure du test (présent dans le nom du fichier)
- Réussite ou Échec du test (présenté sous la forme d'un 1 ou d'un 0)
- RSSI (dans le cas d'une réussite, sinon mettre une colonne vide)
    - La virgule dans la valeur du RSSI peut faire partie du résultat
- Identifiant de la carte (IMEI)

Exemple : Voici le résultat attendu :

```
damien@dgn:~/Desktop/cours/TP$ ./merge_tests_as_csv.sh ./TP1_ex10/*
./TP1_ex10/teraterm_2020-09-09 10-56-10.log
./TP1_ex10/teraterm_2020-09-09 11-01-13.log
./TP1_ex10/teraterm_2020-09-09 11-09-15.log
./TP1_ex10/teraterm_2020-09-09 11-13-14.log
./TP1_ex10/teraterm_2020-09-09 11-15-50.log
./TP1_ex10/teraterm_2020-09-09 11-25-35.log
damien@dgn:~/Desktop/cours/TP$ cat teraterm_output.csv
2020-09-09 10-56-10;0;;864376040797622
2020-09-09 11-01-13;1;8,;864376040770066
2020-09-09 11-09-15;1;6,;864376040871385
2020-09-09 11-13-14;1;4,;864376040778242
2020-09-09 11-15-50;0;;864376040799156
2020-09-09 11-25-35;1;8,;864376040076605
```

Aide
Voici une liste de commandes qui peuvent vous être utiles, vous pouvez en avoir besoin de certaines, comme d'aucune, selon comment vous faites.
Il n'y a pas qu'une solution à ce problème !

Commandes utiles : grep, cut, wc, awk, basename

Attention, les noms des fichiers contiennent des espaces, ce qui peut poser des problèmes. Plusieurs solutions existent pour gérer cela, voici quelques conseils et informations :

- Affichez un maximum de choses dans votre script pour identifier comment se comporte chaque ligne
- Il est possible d'échapper des variables entre 2 guillemets doubles
    - Par exemple : **base=$(basename $file)** peut avoir un comportement différent de **base=$(basename "$file")**
- Il existe une manière de changer le "Input Field Separator" utilisé dans votre script

Encore une fois, il existe plein de façons de faire, rien de ce que j'ai cité au-dessus n'est obligatoirement nécessaire à utiliser, mais vous pourriez en avoir besoin selon la solution que vous apportez.
À vous de voir comment vous faites !