

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 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 "**1-somme.sh**" dans lequel seront définies 2 variables **a** et **b** et afficher la somme de ces 2 variables :

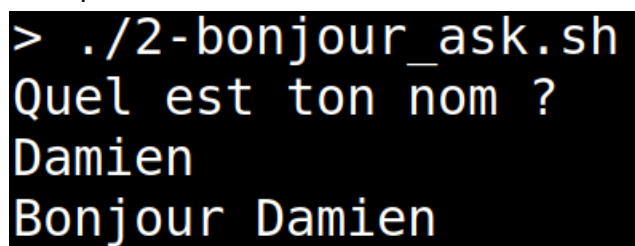
- **a** qui vaut **12**
- **b** qui vaut **30**

Exercice 2

Créez le script "**2-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 :



```
> ./2-bonjour_ask.sh
Quel est ton nom ?
Damien
Bonjour Damien
```

Exercice 3

Créez le script "**3-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 :

```
> ./3-bonjour_param.sh Damien
Bonjour Damien
> ./3-bonjour_param.sh Toto
Bonjour Toto
```

Exercice 4

Créez le script "**4-somme_param.sh**" qui affiche la somme de 2 nombres passés en paramètres du script.

Exemple :

```
> ./4-somme_param.sh 10 11
21
> ./4-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 "**5-somme_param_safe.sh**".

Exemple :

```
> ./5-somme_param_safe.sh
Nombre d'arguments incorrect
> ./5-somme_param_safe.sh 12
Nombre d'arguments incorrect
> ./5-somme_param_safe.sh 12 30
42
```

Exercice 6

Créez le script "**6-for.sh**" qui prend en paramètre un argument "X", et qui affiche les nombres 0 à X sur une nouvelle ligne à l'aide d'une boucle **for**.

Exemple :

```
> ./6-for.sh 4
0
1
2
3
4

> ./6-for.sh 3
0
1
2
3
>
```

Exercice 7

Créez un script "**7-set_binary_perms.sh**" qui prend deux paramètres en entrée :

1. Le chemin d'accès vers un fichier.
2. Trois séquences de trois chiffres binaires chacune, représentant les nouvelles permissions pour le fichier.

Le script devrait effectuer les actions suivantes :

- Vérifier que le script dispose du bon nombre d'arguments
- Appliquer les nouvelles permissions au fichier spécifié
- Afficher un message indiquant que les modifications de permissions ont été effectuées

Assurez-vous que les séquences binaires entrées correspondent aux permissions standard des fichiers Linux (par exemple, 110 pour rw-, 111 pour rwx, etc.).

Exemple :

```
> ls -l myfile
-rw-rw-r-- 1 dg dg 0 déc.  1 09:24 myfile
> ./7-set_binary_perms.sh
Usage: ./7-set_binary_perms.sh <file> <binary_owner> <binary_group> <binary_others>
> ./7-set_binary_perms.sh myfile 111 101 100
Permissions modifiées pour myfile.
> ls -l myfile
-rwxr-xr-- 1 dg dg 0 déc.  1 09:24 myfile
```

Exercice 8

Prenez l'archive "TP_ex8" présente sur l'ENT et extrayez son contenu dans un dossier. Le dossier "TP_ex8" contient plusieurs fichiers, chacun représentant les "logs" d'une journée spécifique. Chaque fichier de log contient des messages associés à un niveau de gravité tels que INFO, WARNING, et ERROR.

Créez un script appelé "**8-filter_logs.sh**" qui prend deux paramètres en entrée :

1. Le chemin d'accès vers le dossier de logs "TP_ex8".
2. Un niveau de gravité (par exemple, INFO, WARNING, ERROR).

Le script doit effectuer les actions suivantes :

- Parcourir le dossier contenant les logs ("TP_ex8" dans votre cas)
- Pour chaque fichier de log, extraire les lignes qui correspondent au niveau de gravité spécifié ou qui sont supérieures à ce niveau
- Afficher dans le terminal pour chaque ligne extraite le jour, la gravité, l'heure et le message

Exemple :

```
> ./8-filter_logs.sh TP1_ex8 INFO
2023-11-27 ERROR 11:20 Une erreur critique est survenue
2023-11-27 ERROR 12:45 Une situation d'urgence a nécessité une réaction immédiate
2023-11-27 WARNING 14:10 Attention, des avertissements persistants
2023-11-27 INFO 15:35 Les opérations se déroulent normalement
2023-11-27 INFO 17:00 Une information importante pour le suivi
2023-11-27 ERROR 18:25 Une erreur grave a interrompu les processus
2023-11-27 ERROR 19:50 Une défaillance critique a nécessité une intervention immédiate
2023-11-27 WARNING 21:15 Attention, des problèmes potentiels ont été identifiés
2023-11-27 INFO 22:40 Le système fonctionne correctement
2023-11-27 INFO 00:05 Les opérations de maintenance sont en cours
2023-11-27 ERROR 01:30 Un incident majeur a interrompu les opérations
2023-11-27 ERROR 02:55 Une défaillance sérieuse a nécessité une intervention immédiate
2023-11-27 INFO 05:45 Les opérations normales ont repris
2023-11-27 INFO 07:10 Une information utile pour les développeurs

> ./8-filter_logs.sh TP1_ex8 WARNING
2023-11-27 ERROR 11:20 Une erreur critique est survenue
2023-11-27 ERROR 12:45 Une situation d'urgence a nécessité une réaction immédiate
2023-11-27 WARNING 14:10 Attention, des avertissements persistants
2023-11-27 ERROR 18:25 Une erreur grave a interrompu les processus
2023-11-27 ERROR 19:50 Une défaillance critique a nécessité une intervention immédiate
2023-11-27 WARNING 21:15 Attention, des problèmes potentiels ont été identifiés
2023-11-27 ERROR 01:30 Un incident majeur a interrompu les opérations
2023-11-27 ERROR 02:55 Une défaillance sérieuse a nécessité une intervention immédiate
2023-11-27 WARNING 04:20 Attention, des anomalies ont été détectées
2023-11-27 ERROR 08:35 Une situation d'urgence s'est produite
2023-11-27 ERROR 10:00 Un incident majeur a nécessité une intervention immédiate
2023-11-27 WARNING 11:25 Attention, des avertissements persistants
2023-11-27 ERROR 15:40 Une erreur grave a interrompu les processus
2023-11-27 ERROR 17:05 Une défaillance critique a nécessité une intervention immédiate
2023-11-27 WARNING 18:30 Attention, des problèmes potentiels ont été identifiés

> ./8-filter_logs.sh TP1_ex8 ERROR
2023-11-27 ERROR 11:20 Une erreur critique est survenue
2023-11-27 ERROR 12:45 Une situation d'urgence a nécessité une réaction immédiate
2023-11-27 ERROR 18:25 Une erreur grave a interrompu les processus
2023-11-27 ERROR 19:50 Une défaillance critique a nécessité une intervention immédiate
2023-11-27 ERROR 01:30 Un incident majeur a interrompu les opérations
2023-11-27 ERROR 02:55 Une défaillance sérieuse a nécessité une intervention immédiate
2023-11-27 ERROR 08:35 Une situation d'urgence s'est produite
2023-11-27 ERROR 10:00 Un incident majeur a nécessité une intervention immédiate
2023-11-27 ERROR 15:40 Une erreur grave a interrompu les processus
2023-11-27 ERROR 17:05 Une défaillance critique a nécessité une intervention immédiate
2023-11-27 ERROR 22:45 Un incident majeur a interrompu les opérations
2023-11-27 ERROR 00:10 Une défaillance critique a nécessité une intervention immédiate
2023-11-27 ERROR 05:10 Une erreur grave a interrompu les processus
2023-11-27 ERROR 07:15 Une situation d'urgence s'est produite
2023-11-28 ERROR 14:52 Un problème critique est survenu
2023-11-28 ERROR 14:53 Une erreur imprévue est apparue
2023-11-28 ERROR 12:05 Un incident majeur s'est produit
2023-11-28 ERROR 13:15 Un dysfonctionnement a été détecté
2023-11-28 ERROR 17:10 Une erreur critique a été détectée
```

Exercice 9

Créez un fichier **"9-config.conf"** avec le contenu suivant :

```
# Configuration du serveur
ServerName example.com
DocumentRoot /var/www/html

# Configuration de la base de données
DBHost localhost
DBUser user
DBPassword secret
```

Créez un script **"9-modify_config.sh"**, qui prend en paramètres 3 valeurs :

- la valeur du ServerName
- la valeur du DBHost
- la valeur du DBUser

L'objectif du script est de créer un fichier nommé **"9-config-ok.conf"** dans lequel les valeurs de **"ServerName"**, **"DBHost"** et **"DBUser"** ont été remplacées par les valeurs passées en paramètres du script.

Exemple :

```
> ./9-modify_config.sh
Usage: ./9-modify_config.sh <ServerName> <DBHost> <DBUser>
> ./9-modify_config.sh site.com 127.0.0.1 damien
> cat 9-config-ok.conf
# Configuration du serveur
ServerName site.com
DocumentRoot /var/www/html

# Configuration de la base de données
DBHost 127.0.0.1
DBUser damien
DBPassword secret
```

Exercice 10

Prenez l'archive "TP_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 SUCCEDED** vers la fin du fichier :

```
[2020-09-09 11:04:50.824]  
[2020-09-09 11:04:50.928] TEST SUCCEDED  
[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]  
[2020-09-09 11:04:21.209] GSM 800 START TASK OK
```

Afin d'obtenir un récapitulatif des différents tests effectués, créez un script

"10-tests_to_json.sh" qui a pour but de créer un fichier JSON qui contient 2 clefs :

- **success**
- **failure**

La clef **success** contiendra comme valeur une liste d'éléments, qui pour chaque élément possède les clefs suivantes :

- **imei** : IMEI de la carte
- **date** : Date et heure du test (présent dans le nom du fichier)
- **rss** : Valeur du RSSI pour ce test (la virgule dans la valeur du RSSI peut faire partie du résultat)

La clef **failure** contiendra comme valeur une liste d'éléments, qui pour chaque élément possède les clefs suivantes :

- **imei** : IMEI de la carte
- **date** : Date et heure du test (présent dans le nom du fichier)

Exemple :

```
> ./10-tests_to_json.sh TP_ex10/*  
Fichier teraterm_output.json créé avec succès.  
>  
> cat teraterm_output.json  
{  
  "success": [  
    {"imei": "864376040770066", "date": "2020-09-09 11-01-13", "rss": "8,"},  
    {"imei": "864376040871385", "date": "2020-09-09 11-09-15", "rss": "6,"},  
    {"imei": "864376040778242", "date": "2020-09-09 11-13-14", "rss": "4,"},  
    {"imei": "86437604076605", "date": "2020-09-09 11-25-35", "rss": "8,"}],  
  "failure": [  
    {"imei": "864376040797622", "date": "2020-09-09 10-56-10"},  
    {"imei": "864376040799156", "date": "2020-09-09 11-15-50"}]  
}
```

Pour une meilleure compréhension de la structure attendue, le fichier "teraterm_output.json" ressemble à ceci une fois formaté :

```
{
  "success": [
    {
      "imei": "864376040770066",
      "date": "2020-09-09 11-01-13",
      "rssi": "8,"
    },
    {
      "imei": "864376040871385",
      "date": "2020-09-09 11-09-15",
      "rssi": "6,"
    },
    {
      "imei": "864376040778242",
      "date": "2020-09-09 11-13-14",
      "rssi": "4,"
    },
    {
      "imei": "864376040076605",
      "date": "2020-09-09 11-25-35",
      "rssi": "8,"
    }
  ],
  "failure": [
    {
      "imei": "864376040797622",
      "date": "2020-09-09 10-56-10"
    },
    {
      "imei": "864376040799156",
      "date": "2020-09-09 11-15-50"
    }
  ]
}
```

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 !