M21
Introduction à la programmation
Travaux Pratiques

Sami Evangelista

IUT de Villetaneuse

Département Réseaux et Télécommunications

2022–2023

http://www.lipn.univ-paris13.fr/~evangelista/cours/M21

Table des matières

Tous les scripts à écrire dans les exercices du module sont à placer dans un répertoire `M21` que vous aurez créé dans votre répertoire personnel. Il est conseillé d'utiliser l'éditeur de texte emacs pour écrire vos scripts (`emacs fichier.sh &` dans le terminal).

Exercice 1 — Un premier script bash

Notre premier script permettra d'afficher un message de bienvenue à l'utilisateur contenant son nom, puis affichera la date courante (commande `date`) et enfin le répertoire de travail (commande `pwd` pour *print working directory*).

1 Créez un fichier `bonjour.sh`.

2 Éditez le, puis entrez le contenu suivant :

```
#!/bin/bash

la_date=date
echo 'Bonjour $USER et bienvenue!'
echo "La date courante est: la_date"
echo "Vous êtes dans le répertoire "
echo pwd
```

(Ce script ne résoud pas notre problème. Nous le le corrigerons par la suite.)

3 Rendez le fichier exécutable (`chmod +x bonjour.sh`) puis exécutez le (`./bonjour.sh`).

On devrait alors observer le résultat ci-dessous :

```
Bonjour $USER et bienvenue!
La date courante est: la_date
Vous êtes dans le répertoire
pwd
```

ce qui ne correspond pas au résultat attendu. En effet, on voudrait observer quelque chose comme :

```
Bonjour toto et bienvenue!
La date courante est: dim. 09 oct. 2022 14:21:50 CEST
Vous êtes dans le répertoire
/home/toto/M21
```

4 Corrigez le programme sans supprimer ni ajouter de ligne pour qu'il affiche bien le résultat attendu.

5 Modifiez enfin votre programme pour qu'il ne crée aucune variable et que le chemin du répertoire s'affiche sur la même ligne et à la suite du message `Vous êtes dans le répertoire`.

Exercice 2 — Utilisation de la variable PATH

Dans l'exercice précédent nous avions, pour exécuter notre script, entré `./bonjour.sh`. Les caractères `./` indiquent à l'interpréteur que le script se trouve dans le répertoire courant (le répertoire `.`). Sans ces deux caractères, l'interpréteur ne saurait pas où trouver le script, car le répertoire courant n'est généralement pas dans la liste des répertoires de la variable d'environnement `PATH`.

Nous allons voir dans cet exercice comment cette variable peut être utilisée.

1 Créez un répertoire `bin` dans votre répertoire personnel.

2 Créez, dans ce nouveau répertoire, un lien vers votre script `bonjour.sh` appelé `bonjour` (sans l'extension `.sh`).

3 Ajoutez ce nouveau répertoire à la liste des répertoires de la variable `PATH`.

4 Pour tester, déplacez vous dans un répertoire quelconque, p.ex. `/tmp`, puis entrez la commande `bonjour`. Ceci devrait exécuter votre script comme prévu.

5 De même, modifiez légèrement le fichier `bonjour.sh` (p.ex., mettez trois points d'exclamation au lieu d'un dans le message de bienvenue) et refaites le test précédent. Vous devriez voir la modification. En effet, les noms `bonjour.sh` et `bonjour` pointent vers le même contenu. Modifier `bonjour.sh` ou `bonjour.sh` revient donc au même.

Remarquez que la modification de la variable `PATH` ne sera effective que pour le processus bash en cours. Par exemple, si vous ouvrez un nouveau terminal ou si vous vous déconnectez puis reconnectez, la modification sera perdue. Pour résoudre ce problème une solution courante est de modifier la variable `PATH` dans le fichier `.bashrc` de votre répertoire personnel et d'exporter ensuite la variable pour que la modification soit visible par toute la descendance du processus. En effet, ce fichier est un fichier lu et exécuté au lancement d'un processus bash et on l'utilise souvent pour définir de nouveaux alias, personnaliser le message d'invite (variable `PS1`), ajouter des répertoires personnels au `PATH`, ...

Exercice 3 — Lecture sur l'entrée standard

Nous allons voir dans cet exercice comment un script peut lire des données depuis l'entrée standard, ce qui pourra correspondre soit à des données saisies au clavier, soit au contenu d'un fichier, soit à la sortie standard d'une autre commande.

1 Écrivez un script `salutation.sh` qui demande à l'utilisateur son nom puis son âge et affiche ensuite un message de salutation, comme ceci :

```
Quel est votre nom ?
Gaston Lagaffe
Quel est votre âge ?
25
Bonjour Gaston Lagaffe! C'est cool d'avoir 25 ans.
```

2 On suppose maintenant que notre script `salutation.sh` sera utilisé de manière non interactive (i.e., qu'un autre script fera appel à lui) et lira le nom et l'âge contenu dans le fichier `entree_salutation` :

```
Gaston Lagaffe
25
```

Donnez deux commandes, l'une utilisant un tube (`|`), l'autre utilisant une redirection (`<`), permettant à `salutation.sh` de lire le contenu de `entree_salutation`.

Exercice 4 — Opérations arithmétiques

Pour résoudre une équation de la forme $ax^2 + bx + c = 0$ on calcule d'abord le discriminant $\Delta = b^2 - 4ac$. Le but de cet exercice est d'écrire trois scripts qui effectuent ce calcul et qui se différencient par la manière dont ils reçoivent des données en entrée.

1 Écrivez un script `discriminant.sh` qui lit successivement sur l'entrée standard a, b, puis c et affiche son discriminant. Voici un exemple d'exécution du script :

```
Entrez la valeur de a
2
Entrez la valeur de b
8
Entrez la valeur de c
1
Le discriminant vaut 56.
```

2 Écrivez une deuxième version du script, nommée `discriminant_une_ligne.sh`, qui lira les valeurs de a, b et c sur la même ligne, comme dans l'exemple ci-dessous :

```
Entrez la valeur de a, b et c
2 8 1
Le discriminant vaut 56.
```

3 Enfin, écrivez une troisième version du script, nommée `discriminant_args.sh` avec laquelle les valeurs de a, b et c ne seront plus lues sur l'entrée standard mais passées en argument du script, comme dans l'exemple ci-dessous :

```
$ discriminant_args.sh 2 8 1
Le discriminant vaut 56.
```

Exercice 5 — Analyse d'un chemin absolu

Le but de cet exercice est d'étudier quelques fonctionnalités de bash permettant de manipuler les chaînes de caractères.

Vous écrirez un script `analyse_chemin.sh` qui prend en argument le chemin absolu d'un fichier et affiche diverses informations. Le fichier n'existe pas nécessairement. Voici un exemple d'exécution du script :

```
$ analyse_chemin.sh /home/gaston/documents/notes.txt
Le fichier s'appelle notes.txt
Il se trouve dans le répertoire /home/gaston/documents
Son extension est txt
Il se trouve à une profondeur de 3
```

Remarques :

— On supposera que le chemin passé en argument ne contient pas de blanc et que le fichier à toujours une extension (i.e., se termine par quelque chose de la forme `.ext`).
— La profondeur est le nombre de répertoires qui se trouvent sur le chemin vers le fichier. Ici on en a 3 : `home`, `gaston`, et `documents`.

Exercice 6 — Test de la nature d'un fichier

Écrire un script `test_fichier.sh` prenant un unique argument que nous noterons `f` et qui affichera un des messages ci-dessous :
— `f existe et c'est un fichier ordinaire`
— `f existe et c'est un répertoire`
— `f existe mais ce n'est ni un répertoire ni un fichier ordinaire`
— `f n'existe pas`
selon la nature de `f`. Le script doit au préalable tester s'il a reçu un (et un seul) argument. Dans le cas contraire il affichera un message d'erreur (sur la sortie erreurs) et terminera avec le code 1.

Dans les exercices suivants, vous ajouterez toujours en début de script les instructions permettant de tester que le script a été correctement exécuté et de l'arrêter prématurèment dans le cas contraire.

Exercice 7 — Ouverture d'un fichier

Écrire un script `ouvre_fichier.sh` qui prend un unique argument qui est soit le nom d'un fichier soit une URL (de type http ou https). Selon l'argument reçu, le script :
— ouvrira le fichier avec un lecteur PDF (evince, xpdf, ...) si c'est un fichier d'extension pdf ;
— ouvrira le fichier avec un lecteur d'images (emacs, ...) si c'est une fichier d'extension png ou jpg ;
— ouvrira le fichier avec un navigateur web (firefox, ...) si c'est une URL http ou https (i.e., s'il commence par `http://` ou `https://`) ;
— ou affichera un message d'erreur (*Je ne sais pas comment ouvrir ce fichier.*) dans tout autre cas.

Exercice 8 — Somme d'entiers

Écrire un script `somme.sh` qui prend en arguments une liste d'entiers puis calcule et affiche la somme de ces entiers. Si aucun argument n'est fourni, le script affichera `0`. Par exemple :

```
$ somme.sh
0
$ somme.sh 2 4 8 16
30
```

Le script ne devra pas vérifier que les arguments fournis sont bien des entiers.

Exercice 9 — Conversion de fichiers

L'outil `convert` permet de convertir des fichiers d'image dans de nombreux formats (p.ex., jpg, png, pdf). Il s'utilise simplement depuis la ligne de commande en lui fournissant deux arguments : le chemin du fichier à convertir et le chemin du fichier en sortie (obtenu après conversion). L'outil se base sur l'extension du fichier en sortie pour déterminer son format. Par exemple, la commande ci-dessous convertit le fichier `background.jpg` au format png :

```
$ convert background.jpg background.png
```

Après conversion, on se retrouve avec deux fichiers :

```
$ ls background.*
background.jpg  background.png
```

En utilisant cette commande, écrire un script `convertit_fichiers.sh` prenant 3 arguments :
— le chemin d'un répertoire existant et modifiable (noté `rep`) ;
— une extension source (notée `src`) ;
— et une extension destination (notée `dst`).
Le script convertira chaque fichier du répertoire `rep` ayant l'extension `src` en un fichier de même nom mais ayant l'extension `dst`. Un message sera affiché pour chaque fichier converti.

Voici un exemple d'exécution du script :

```
$ ls images/
background.jpg  cv.jpg  photo.png
$ ./convertit_fichiers.sh images jpg png
images/background.jpg -> images/background.png
images/cv.jpg -> images/cv.png
$ ls images/
background.jpg  background.png  cv.jpg  cv.png  photo.png
```

Exercice 10 — Découverte d'hôtes

Écrire un script `decouverte24.sh` prenant en argument un identifiant de réseau sur 3 octets (donc un réseau en /24) et qui enverra une unique demande d'écho (i.e., un message ping) à toutes les IPs de ce réseau. Le script affichera les IPs de toutes les interfaces ayant répondu à la demande.

Voici un exemple d'exécution du script :

```
$ ./decouverte24.sh 192.168.15
192.168.15.24 a répondu
192.168.15.82 a répondu
192.168.15.88 a répondu
192.168.15.115 a répondu
192.168.15.254 a répondu
```

Remarques :

— Utiliser la commande `seq` (consulter le manuel) combinée avec une substitution de commande (`$(seq ...)`) pour pouvoir itérer sur une liste d'entiers.
— C'est l'option `-c1` de la commande `ping` qui permet d'envoyer une seule demande. Sinon la commande envoie des demandes en continu.
— Limitez vos tests à quelques adresses seulement (5, par exemple) pour que le script termine en un temps raisonnable.
— Attention à ne pas envoyer de demande sur une adresse de réseau ou sur une adresse de diffusion.

Exercice 11 — Analyse de fichiers journaux

Les fichiers journaux (ou log) contiennent des informations importantes sur des événements survenus (p.ex., l'arrêt du système, le démarrage d'un programme). Sous Linux, on les trouve généralement dans le répertoire `/var/log`. On se propose dans cet exercice d'écrire un script permettant d'extraire d'un journal centralisant des événements relatifs à plusieurs machines ceux d'une machine en particulier. Les lignes du journal ont la structure suivante :

```
date-et-heure;adresse-ip;événement
```

Voici, par exemple, un contenu concret :

```
20221031 10:00:02;10.10.0.1;démarrage du système
20221031 10:00:08;10.10.0.1;démarrage du service d'impression
20221103 22:18:54;10.20.0.2;désactivation de l'interface eth2
20221104 02:12:44;10.20.0.3;arrêt du système
```

Écrivez un script `extrait_fichier_log.sh` prenant en argument le chemin d'un fichier de journal et une adresse IP et qui affichera sur l'entrée standard les lignes du fichier concernant l'adresse IP passée en argument uniquement (i.e., celle dont la deuxième colonne contient l'adresse passée en argument).

Exercice 12 — Analyse de répertoire de journaux

On reprend dans cet exercice la structure des journaux de l'exercice précédent.

Écrivez maintenant un script `extrait_repertoire_log.sh` prenant en argument le chemin d'un répertoire et une adresse IP et qui affichera sur l'entrée standard les lignes de tous les fichiers journaux du répertoire concernant l'adresse IP passée en argument. On considérera que les journaux ont l'extension `.log`.

Exercice 13 — Tri de fichiers de journaux

Toujours avec la même structure de fichier que dans l'exercice 11, écrivez un script `tri_log_fichier.sh` prenant en argument le chemin d'un fichier de journal et qui triera les événements dans différents fichiers. On suppose que le réseau est composé de plusieurs sous-réseaux en /16 et on souhaite que chaque événement relatif à une IP soit écrit dans

un fichier contenant les événements du réseau de cette IP. En supposant que le script a analysé le fichier `journal.log`, les événements relatifs à une IP du réseau A.B.0.0 devront être rangés dans le fichier `journal-A.B.log`.

En reprenant l'exemple de l'exercice 11, l'éxecution de la commande `tri_log_fichier.sh journal.log` produira deux fichiers `journal-10.10.log` et `journal-10.20.log` contenant respectivement :

```
20221031 10:00:02;10.10.0.1;démarrage du système
20221031 10:00:08;10.10.0.1;démarrage du service d'impression
```

et

```
20221103 22:18:54;10.20.0.2;désactivation de l'interface eth2
20221104 02:12:44;10.20.0.3;arrêt du système
```

Exercice 14 — Analyse de fichiers HTML

Écrivez un script `telecharge_liens.sh` prenant deux arguments : l'URL d'une page HTML, et un type de fichier (`pdf`, `jpg`, `png`, ...). Le script téléchargera la page passée en premier argument (avec `curl` ou `wget`) puis téléchargera dans le répertoire courant tous les fichiers du type passé en deuxième argument et référencés dans la page.

On s'intéressera uniquement au fichiers référencés dans le code HTML à l'aide de balises `a` (les liens hypertexte) ou `img` (les images), soit par exemple :

```
<a href="document.pdf"/>
<img src="photo.jpg"/>
<img src="http://www.exemple.com/background.jpg"/>
```

Notez que les chemins indiqués dans les balises peuvent être soit absolus (i.e., commencer par `http://` ou `https://`) soit relatifs.

On supposera pour simplifier qu'il ne peut y avoir qu'une seule balise `a` ou `href` par ligne. S'il y en a plusieurs, le script téléchargera uniquement le premier fichier référencé. Dans l'exemple ci-dessous, le fichier `test.png` ne sera donc pas téléchargé :

```
<a href="img.png"/> <img src="test.png"/>
```

On supposera également que l'attribut `href` ou `src` se trouve sur la même ligne que la balise. Le script pourra donc ignorer les codes HTML suivant :

```
<img
src="img.png"/>
<a
href="test.png"/>
```

Par contre, la balise peut se trouver n'importe où dans une ligne. Par exemple, le code HTML suivant devra être traité correctement par votre script :

```
lien vers un document: <a href="document.pdf"/>
```