TP 7 - Scripts `bash` plus avancés

Consignes de rendu

1. Créez le dossier `~/Linux/TP07` et placez-vous dedans.
2. Créez-y un fichier `reponses.txt` pour noter vos observations.
3. Le TP est à rendre par e-mail en suivant le format habituel (voir slides du premier cours). Pour obtenir l'archive à envoyer depuis le répertoire `~/Linux/`, utilisez :

   `tar jcvf numéro_d'étudiant-prénom français-TP07.tar.bz2 TP07`

Fonctions basiques

Exercice 1 (Archiver)

La commande `tar` utilisée pour archiver votre travail est longue à retenir et à écrire. Écrivez une fonction `archiver` qui prend en paramètre le nom d'un répertoire et crée une archive respectant les conventions de nommage habituelles. Par exemple, pour le numéro d'étudiant `1234567` et le prénom `Pierre` :

```
$ ls
TP01
$ archiver TP01
$ ls
TP01
1234567-Pierre-TP01.tar.bz2
```

Placez la fonction dans `~/.bashrc` pour pouvoir la réutiliser dans de nouveaux terminaux. Comment la rendre disponible immédiatement dans le terminal courant ?

Exercice 2 (Sauvegarder ses fichiers)

Écrivez une fonction permettant de sauvegarder facilement des fichiers importants.

(a) Créez un répertoire `~/sauvegardes` pour conserver vos sauvegardes.

(b) Écrivez une fonction `sauvegarder` qui reçoit le chemin d'un fichier et le copie dans ce répertoire. Placez-la dans `~/.bashrc`. Par exemple, `sauvegarder ~/.bashrc` doit créer une copie de ce fichier dans `~/sauvegardes`.

(c) Modifiez la fonction pour éviter qu'une sauvegarde n'écrase la précédente : ajoutez la date et l'heure au nom de chaque copie. **Indication :** consultez `date` et son format `+%Y%m%d-%H%M%S`. Deux sauvegardes effectuées dans la même seconde peuvent encore porter le même nom.

(d) Utilisez un tableau et une boucle pour sauvegarder successivement `~/.bashrc` et `~/.profile`.

Fonctions avec tableaux

Exercice 3 (Tous les couples différents)

Écrivez une fonction `tous_couples_differents` qui prend en paramètre un tableau

de chaînes et affiche tous les couples différents de valeurs de ce tableau. Par exemple :

```
$ mon_tableau=("bonjour" "gnu" "linux")
$ tous_couples_differents "${mon_tableau[@]}"
bonjour gnu
bonjour linux
gnu linux
```

Chaque couple doit être affiché une seule fois, sans afficher les couples d'un élément avec lui-même. Si le tableau contient des valeurs répétées, des lignes identiques peuvent apparaître.

Attention : si vous décidez d'affecter un nom à votre variable dans la fonction pour que le code soit plus lisible, il faut utiliser la syntaxe vue au cours pour récupérer tous les éléments du tableau :

```
tab=("$@")  # dans la fonction, pour utiliser tab au lieu de $1
```

Indication : pour parcourir des indices, vous pouvez utiliser une boucle arithmétique Bash (syntaxe proche du C) :

```
for ((i=0; i<n; i++)); do
    echo "$i"
done
```

Pour éviter les doublons, le second indice peut commencer à `i+1`.

Scripts et paramètres

Exercice 4 (Un script pour les générer tous)

Nous allons écrire un script `nouveau_script.sh` qui génère un nouveau script prêt à l'emploi, c'est-à-dire un fichier vide et exécutable avec l'extension `.sh` contenant :

1. le bon entête,
2. une description,
3. le nom de l'auteur, et son adresse e-mail.

Par exemple, exécuter la commande `./nouveau_script.sh essai.sh` créera le fichier exécutable `essai.sh` contenant :

```
#!/usr/bin/env bash
#
# Description du script
#
# Utilisation:
#
# Copyright (C) 2026
#
# Auteurs: Anthony Labarre <anthony@debian>
```

(a) Écrivez une première version la plus générique possible à l'aide des variables d'environnement. Ainsi, si vous envoyez `nouveau_script.sh` à un ami, le script utilisera automatiquement les coordonnées de votre ami. L'adresse e-mail sera au format "`nom d'utilisateur@nom de la machine`".

(b) Une fois que votre script fonctionne, modifiez-le pour qu'il ne fasse rien et prévienne l'utilisateur d'une erreur si le nom du fichier à créer est vide. On peut vérifier si une chaîne est vide avec `if [[ -z $chaine ]]`.

(c) Ensuite, modifiez-le pour qu'il rajouter un `.sh` à la fin du nom du fichier si l'utilisateur l'a oublié.

(d) Enfin, demandez une confirmation à l'utilisateur si un fichier portant le nom demandé existe déjà.

Exercice 5 (Installation)

Prérequis : la commande `apt-file` doit être installée et son index mis à jour (voir `apt-file update`). Les résultats dépendent des dépôts configurés et de la version de Debian : les exemples ci-dessous sont indicatifs.

Écrivez un script `installer.sh` qui prend en paramètre une chaîne et installe automatiquement le paquet contenant le fichier correspondant. Par exemple :

```
$ ./installer.sh gentoopenguin.png
Recherche en cours, un instant...
Il y a exactement un paquet contenant ce nom: gnome-devel-docs
Voulez-vous l'installer? [o/n]
```

L'installation sera lancée si l'utilisateur tape `'o'`, ou annulée s'il tape `'n'`. Si le fichier est introuvable, ou s'il y a plus d'un paquet correspondant, on ne fait rien. Par exemple :

```
$ ./installer.sh jenexistepas
Recherche en cours, un instant...
Aucun résultat trouvé; utilisez sudo apt-file update si nécessaire.
$ ./installer.sh beamer.sty
Recherche en cours, un instant...
Plusieurs résultats trouvés, c'est trop pour moi.
```

Défis (facultatifs)

Exercice 6 (Installation avec choix multiples (approfondissement facultatif))

Modifiez votre script de l'exercice 5 pour obtenir un script `installer_multi.sh` qui gère les cas où plusieurs paquets correspondent. Par exemple :

```
$ ./installer_multi.sh beamer.sty
Recherche en cours, un instant...
Plusieurs résultats trouvés (exemple):
    [0] texlive-latex-extra
    [1] texlive-pictures
    [2] texlive-publishers
    [3] texlive-xetex
Entrez le numéro du paquet à installer (ou n pour annuler):
```

On triera les paquets dans l'ordre alphabétique.