Utilisation du shell Unix
7 - Premiers scripts

1 Création d'un répertoire pour ranger vos scripts

Créez un répertoire `bin` dans votre répertoire de travail par défaut. Positionnez-vous dans `bin` et créez, avec votre éditeur favori, un fichier `essai` qui affiche ses deux premiers paramètres sur 2 lignes séparées.
Exemple :

```
essai un deux
```

doit afficher :

```
Param 1 : un
Param 2 : deux
```

Rendez-ce fichier exécutable et testez-le avec 1, 2 ou 3 paramètres et observez ce qui se passe.

Pour répondre aux questions posées dans le sujet, créez dans votre répertoire `bin` un fichier `CR7.txt` que vous modifierez avec votre éditeur favori. Mettez simplement la réponse aux questions en texte simple, inutile de vous fatiguer à faire des copies d'écran.

Pour que vous puissiez utiliser `essai` (et tous les autres scripts du répertoire `bin`) depuis n'importe quel répertoire, il faut ajouter le répertoire `bin` que vous venez de créer dans la liste des répertoires où le système va chercher les commandes (`PATH`), et ce de manière permanente.

Pour ce faire il faut ajouter la commande de modification du `PATH` dans votre fichier `.profile`, qui est exécuté à l'ouverture de session.
Ouvrez votre fichier `.profile`. Vous pouvez observez que des commandes sont déjà écrites pour ajouter le répertoire `bin` à votre `PATH` (en principe lignes 19 à 22).

⟹ **Expliquez le fonctionnement de ces commandes (dans CR7.txt).**

Pour vérifier que ces commandes fonctionnent, vous pouvez fermer votre session et la ré-ouvrir ou plus simplement lancer un shell en mode « session » (le fichier `.profile` est exécuté) avec la commande :
`bash --login` ou plus simplement `bash -l`
Lancez `essai` dans ce shell pour vérifier que ça fonctionne.

Vous disposez maintenant d'un répertoire pour ranger vos commandes personnelles, répertoire qui vous est accessible quel que soit votre répertoire de travail.

2 Substitutions

Certains caractères spéciaux sont interprétés par le shell qui *substitue* dans une commande la chaîne contenant ces caractères par le résultat de l'interprétation. Quand on utilise par exemple la commande : `echo $HOME`, le shell commence par remplacer la chaîne `"$HOME"` par la valeur de la variable **AVANT** d'exécuter la commande `echo`.

De même, avec la commande : `ls *.c`, le shell remplace `"*.c"` par la liste des fichiers dont le nom correspond dans le répertoire courant **AVANT** de lancer la commande `ls`. Celle-ci reçoit donc en paramètre la liste des fichiers.

⟹ **Essayez les commandes suivantes et expliquez ce qui se passe (dans CR7.txt) :**

```
essai M*
essai $HOME $PATH
essai '$HOME' '$PATH'
```

On peut aussi demander au shell de remplacer une commande par son résultat (ce qu'elle affiche). Cela permet de passer le résultat d'une commande en paramètre d'une autre commande. Il suffit pour cela de noter la commande entre `$(...)` : `$(`*`commande`*`)`

Essayez la commande `date`. Essayez ensuite : `essai $(date)`
$\Longrightarrow$ **Expliquez ce qui se passe pour cette dernière commande (dans CR7.txt).**

3 Combien d'utilisateurs ?

La commande `who` affiche la liste des utilisateurs connectés (1 par ligne). La commande `wc` compte les lignes, les mots et les caractères contenus dans les fichiers passés en paramètre ou sur son entrée standard si aucun fichier n'est passé.

Essayez `who`. Réessayez `who` en redirigeant sa sortie sur l'entrée de `wc` (avec un *tube*).

En utilisant `who` en conjonction avec `wc` (faire `man wc` pour déterminer la bonne option), créez dans votre répertoire `bin` une commande `hmu` qui affiche :

`Il y a` *`XX`* `utilisateurs connectés.`

`XX` est bien sûr le nombre d'utilisateurs effectivement connectés.

L'intérêt de cette commande est limité sur les machines Linux des salles de TP puisque vous êtes le seul utilisateur. Pour bien tester cette commande, connectez-vous au serveur `gigondas` :

```
ssh gigondas
```

Répondez `yes` à la première question (il ne la pose que lors de la première connexion), puis saisissez votre mot de passe de session (attention, pas d'écho du mot de passe).

Vous avez pu observer que `who` affiche plusieurs fois le nom d'un utilisateur (s'il a plusieurs fenêtres ouvertes). En vous aidant des commandes `cut` et `sort`, modifiez votre commande `hmu` pour qu'elle n'affiche que le nombre d'utilisateurs différents.

Pour tester cette dernière version, vous pouvez vous connecter deux fois à `gigondas`, depuis deux fenêtres différentes, puis lancer `hmu` sur `gigondas`.

4 Soyons prudents...

Unix est parfois un peu violent et ne donne pas droit à l'erreur. Ainsi, un fichier détruit par la commande `rm` ne peut pas être récupéré. Pour éviter ça, on va écrire une commande `del` qui envoie les fichiers passés en paramètre dans un répertoire (baptisé `Poubelle`) de votre répertoire de travail par défaut (`HOME directory`). On utilisera `del` pour effacer des fichiers, ainsi en cas de fausse manœuvre, on pourra récupérer les fichiers dans le répertoire `Poubelle`. De temps à autre, on ira vider la poubelle avec la commande `rm`.

Revenez dans votre répertoire *HOME* et créez le répertoire `Poubelle`.

Créez un fichier `del` dans votre répertoire `bin`. Ce fichier devra contenir un script permettant de déplacer (commande `mv`) **tous les fichiers** passés en paramètre dans le répertoire `Poubelle`. Vous prendrez garde à ce que ce script fonctionne bien quel que soit le répertoire de travail au moment de l'appel.

Note : les variables `$@` ou `$*` contiennent **tous les paramètres** passés au script.

Rendez le fichier `del` exécutable et testez-le, en commençant par des fichiers bidons, créés pour l'occasion. Vous pouvez pour ce faire utiliser la commande `touch` qui simule la modification d'un fichier (en changeant la date de dernière modification) si le fichier existe, ou qui crée un fichier vide s'il n'existe pas. Exemple : `touch titi toto`

5 Récupération d'une archive dans `Public`

Pour les TPs (système) on vous demande souvent de récupérer une archive dans le répertoire `Public` puis de l'extraire. On souhaite écrire un script baptisé `recup` qui simplifie ce travail. Par exemple pour récupérer et extraire l'archive `shell.tgz` du répertoire `Public`, on voudrait ne taper que : `recup shell`

Programmez ce script.

6 Ajoutons un mode d'emploi

Il est souvent pratique d'afficher un mode d'emploi lorsque l'utilisateur d'un script se trompe dans les paramètres. Ainsi le script précédent ne fonctionnera pas si on le lance sans paramètre. On voudrait donc afficher dans ce cas un mode d'emploi.

Modifiez le script précédent pour qu'il fonctionne comme suit :

```
recup
```

affichera :

```
Usage : recup fichier (.tgz)
```

tandis que :

```
recup shell
```

effectuera le travail prévu à la section précédente.

Notes :

- on peut tester si une variable est vide avec :
  `test -z "$variable"` ou avec `test "$variable" == ""`
- la syntaxe du *si-alors-sinon* en shell est la suivante :

```
if commande
then
   commande
   ...
else
   commande
   ...
fi
```

7 Compte-rendu

Vous devez rendre à l'enseignant un listing [1] de tous les scripts que vous avez écrits ainsi que les réponses aux questions des sections 1 et 2. Pour cela utilisez le programme `imprime` (qui est un script shell écrit spécialement pour faciliter les comptes-rendus, voir `/soft/bin/imprime`).

Ce programme ajoute les numéros de ligne et optimise le nombre de pages imprimées.

Placez-vous dans votre répertoire `~/bin` puis lancez `imprime` avec par exemple :

```
imprime CR7.txt essai hmu del recup
```

C'est beau non ?

Si vous ne pouvez pas imprimer, enregistrez une copie du fichier PDF produit par la commande `imprime` puis envoyez-la par courriel à l'enseignant en ayant soin de nommer le fichier avec vos noms.

1. les scripts imprimés sur papier