TP2 : Mes Premiers Scripts

Avant propos

- Tous les TPs sont a réaliser **individuellement**.
- Tous les TPs seront à déposer sur claroline **au plustard** à la date indiquée sur claroline (*deadline dure, tous les TPs rendus après cette date ne seront pas considérés).*
- Tous vos rendus devront être sous la forme d'une archive de type *tar.gz* et le nom du rendu sera TPX_NOM_PRENOM.tar.gz où X est le numéro du TP.
- Votre archive devra contenir l'intégralité des scripts créés et s'il y a lieu les commandes tapées et les réponses aux questions dans un fichier `cmd_tpX.txt` où `X` est le numéro du TP.
- **Un TP non rendu impliquera la défaillance à la partie pratique de l'UE**.
- Je vous rappelle que le plagiat est interdit et qu'il est soumis à de **lourdes sanctions disciplinaires** : Ne vous appropriez pas le TP/travail de l'un de vos camarades ou d'une source extérieure ! (Rappel : l'université dispose d'outils pour détecter automatiquement ce genre de fraudes)

Note sur Emacs

Emacs possède un mode d'édition des scripts (avec indentation automatique, coloration de la syntaxe). Ce mode est activé (indiqué par `(Shell-script[bash])` dans la barre d'état) lorsque l'on charge un fichier commençant par la ligne

```
#!/usr/bin/env bash
```

ou avec la combinaison suivante :

```
[alt]-x shell-script-mode
```

1 Bonjour

1. Créez un premier script qui doit afficher `Bonjour à tous!`. Vous nommerez le script `bonjour.sh`. Rendez le script executable et lancez-le dans un terminal.
   Attention : tous vos scripts doivent commencer par la ligne `#!/usr/bin/env bash`.
2. Améliorez le script pour qu'il affiche `Bonjour [Login]!` où `[Login]` est remplacé par votre login (**N.B.** : votre login est stocké dans une variable d'environnement). Vous nommerez le script `bonjour.sh`.

2 Arguments

Le comportement d'un script dépend souvent de données extérieures. Ces données peuvent provenir de plusieurs sources : d'un passage d'arguments du script, d'une interaction avec l'utilisateur lui demandant une saisie manuelle, d'un fichiers, etc.

2.1 Retour sur TD

1. Écrivez un script dans un fichier `param` qui produit l'affichage suivant :

   ```
   La commande tapée est : [la_commande]
   Il y a [nb_param] paramètre/s
   Le premier paramètre est [1er_param]
   Le deuxième paramètre est [2eme_param]
   Le troisième paramètre est [3eme_param]
   La liste de tous les paramètres est [liste_des_param]
   ```

2. Lancez votre script en tapant

   ```
   ./param p1 p2 p3 -8 --help 12 toto
   ```

3. Créez un lien symbolique (commande `ln -s`) de nom `new_param` sur le fichier `param` et essayez de nouveau le script avec

   ```
   ./new_param p1 p2 p3 -8 --help 12 toto
   ```

   Qu'est ce qui change ?

2.2 Manipulation des arguments

Veuillez rendre un script par question : `X_bonjour.sh`, où `X` correspond au numero de la question.

1. Modifiez votre script `bonjour.sh` pour qu'il complète l'affichage produit avec un argument passé sur la ligne de commande. Par exemple :

   ```
   >./1_bonjour.sh Yoda
   Bonjour Yoda !
   ```

2. Modifiez votre script afin qu'il prenne en compte non plus un argument, mais tous les arguments donnés sur la ligne de commande. Par exemple :

   ```
   >./2_bonjour.sh Yoda Luke Rey
   Bonjour Yoda !
   Bonjour Luke !
   Bonjour Rey !
   ```

   OU

   ```
   >./2_bonjour.sh "Baby Yoda" "Rey" "Luke Skywalker"
   Bonjour Baby Yoda !
   Bonjour Rey !
   Bonjour Luke Skywalker !
   ```

3. Modifiez votre script afin qu'il affiche la chaîne "Que la force soit avec vous !" si aucun argument n'a été donné.

4. Modifiez votre script afin qu'il ne salue plus que une personne sur deux. Par exemple

   ```
    >./4_bonjour.sh Yoda Anakin Rey Ben Luke
   Bonjour Yoda !
   Bonjour Rey !
   Bonjour Luke !
   ```

3 Conditionnelle

1. Écrivez un script `division` qui prend 2 paramètres de type entier et qui affiche le résultat de la division du premier par le second. Si le nombre de paramètre est différent de 2 ou si le deuxième paramètre vaut 0, le message suivant doit être affiché :

   ```
   usage : division <entier1> <entier2_non_nul>
   affiche le quotient de <entier1> par <entier2>
   ```

2. Ecrivez un script `frequence` qui prend un nom de fichier en paramètre et qui affiche la fréquence de chacun des mots du texte contenu dans le fichier (comme durant le stage de rentrée). Avant de calculer les fréquences, ce script devra extraire dans le fichier fourni les lignes situées entre "DEBUT DU FICHIER" et "FIN DU FICHIER". On utilisera `grep` et `cut` pour extraire les numéros de ces lignes et les stocker dans des variables debut et fin. Ce script utilisera aussi le script `extract` que vous avez écrit lors du stage de rentrée [1]

3. Modifiez le script précédent pour qu'il affiche un message d'erreur s'il ne trouve pas les deux lignes délimitant le texte ou s'il trouve plus d'une de chacune de ces lignes.

4 Boucles

1. Écrivez un script `factorielle` qui prend un entier `n` en paramètre et qui affiche le résultat de `n`!.
2. Écrivez un script qui calcule la somme de tous les paramètres.
3. Écrivez un script `efface` qui prend une liste de noms de fichier en paramètre et qui fonctionne comme l'effacement des fichiers sous Windows. C'est-à-dire dire que ce script doit déplacer les fichiers dans un répertoire `$HOME/corbeille` situe dans le répertoire personnel de l'utilisateur au lieu de réellement les effacer.
4. Quel problème peut se poser avec le script `efface` ? Comment le résoudre ?

1. Votre script `extract n1 n2 fich` doit afficher les lignes `n1` à `n2` (incluses) du fichier `fich`. Le script doit vérifier que `n1` et `n2` sont positifs, que `n1`$\leq$`n2` et que le fichier existe et est lisible (les commandes `head` et `tail` peuvent aider).