Initiation à l'automatisation des tâches (LI218)
TME Solo – Durée: 1 heure

Thibaut Verron

16 octobre 2013

Documents autorisés

Tous les documents sont autorisés.

Fichiers à récupérer

Vous trouverez des fichiers d'exemple dans le répertoire `/Infos/lmd/2013/licence/ue/li218-2013oct/tmesolo/Groupe4` Aucun des scripts demandés ne vous demandera de modifier ces fichiers, vous pouvez donc simplement travailler en déclarant le chemin avec `export EX="/Infos/lmd/2013/licence/ue/li218-2013oct/tmesolo/Groupe4"` et en utilisant ensuite `$EX/fichier` pour pointer vers les fichiers. Exemple :

```
% ./moyenne.sh LI218 $EX/liste1
40
```

Écriture de vos scripts

Vous pouvez composer vos scripts où vous voulez.

Les scripts que vous aurez à écrire ne doivent pas dépendre du répertoire où ils sont placés ; ils ne doivent pas contenir de référence à des chemins absolus et doivent pouvoir être exécutés sans problème depuis tout répertoire.

N'oubliez pas de rendre vos scripts exécutables par `chmod +x` *`script`* et en mettant le shebang fonctionnel `#!/bin/bash` en début de fichier. N'oubliez pas que sous `emacs`, en mode `shell-script` (automatique dans un fichier en `.sh` ou commençant par `#!/bin/bash`), vous pouvez réaliser ces deux opérations simplement avec la commande `C-c :`

Aucun de vos scripts ne doit laisser après son exécution de fichier non prévu.

Conseils généraux

Le sujet est *volontairement trop long* ! Il est inutile d'espérer tout finir, et il est de loin préférable d'avoir un seul script qui marche, plutôt que d'avoir essayé de toucher à tout et de n'avoir rien qui marche à la fin.

Les questions sont indépendantes, n'hésitez pas à commencer par ce qui vous paraîtra le plus facile.

N'oubliez pas de tester vos scripts ! Vous avez des fichiers d'exemple à votre disposition, vous avez des exemples textuels dans le sujet que vous pouvez copier/coller (le sujet est disponible au format `.pdf` dans le dossier des exemples), et vous pouvez si ça ne suffit pas créer vos propres exemples de toutes pièces.

Si votre script ne fonctionne pas, pensez à l'option `-x` de `bash` qui permet de tracer l'exécution du script (préfixez votre script par `bash -x`).

De même si vous voulez savoir si un script est syntaxiquement correct avant d'essayer de l'exécuter, pensez à l'option `-n` de `bash` : `bash -n` *`script`* vous affiche les erreurs de syntaxe de *`script`* sans l'exécuter et le code de retour de cette instruction est nul si et seulement si *`script`* est syntaxiquement correct.

Vous trouverez dans le fichier d'exemple un script `test.sh` qui applique les exemples de l'énoncé à vos scripts : vous avez simplement à regrouper les scripts dans un même dossier et donner ce nom de dossier comme argument. Ceci est un test minimal que doivent passer vos scripts, cela ne vous dispense en aucun cas de réaliser vos propres tests.

Conventions typographiques et résultats contractuels

Nous utilisons dans ce sujet la convention typographique utilisée dans le cours pour les exemples : les lignes en gras commençant par `%` correspondent à des commandes, le `%` ne faisant pas partie de la commande (c'est le prompt du shell dans notre convention). Les lignes qui suivent (jusqu'à une nouvelle ligne en gras ou la fin du paragraphe) sont le résultat de l'exécution de la commande.

Rendu du TME

Placez vos scripts (et uniquement vos scripts, pas les fichiers d'exemples !) dans un seul répertoire, et lancez le script `rendu.sh` que vous trouverez dans le fichier d'exemples, avec comme argument le chemin de ce répertoire. Si vous le souhaitez, vous pouvez également inclure un fichier texte appelé `README`, où vous pourrez mettre tout commentaire que vous souhaiteriez que je lise (difficultés rencontrées, etc.). N'oubliez pas également que vous pouvez placer des commentaires dans vos scripts.

Par exemple, si vous placez vos scripts dans le dossier `~/TME_solo`, vous lancerez la commande
`/Infos/lmd/2013/licence/ue/li218-2013oct/tmesolo/Groupe4/rendu.sh ~/TME_solo`

Pensez à bien vérifier que tous vos scripts sont bien présents dans le répertoire avant de lancer la commande. En cas d'erreur, n'hésitez pas à la relancer, seule la dernière soumission sera corrigée.

En cas de problème avec le script `rendu.sh`, mettre les scripts dans une archive `<numéro-étudiant>.tar` et envoyer l'archive par mail à l'adresse `thibaut.verron@lip6.fr`.

1 Gestion d'unités d'enseignement

Dans tout le sujet, on travaille avec un fichier d'effectifs d'UE, dont les lignes sont sous la forme

```
code UE;effectif total;nombre de groupes;effectif des groupes
```

Par exemple, le fichier `liste1` contient les lignes

```
LI218;200;5;42;40;38;35;45
LI215;50;2;10;40
LI315;50;3;9;9;32
```

ce qui désigne l'UE LI218, effectif total 200 élèves, séparés en 5 groupes, d'effectifs 42, 40, 38, 35 et 45 élèves, ainsi que l'UE LI215, effectif total 50 élèves, séparés en 2 groupes de 10 et 40 élèves, etc.

Pour toutes les questions à l'exception de la dernière, on supposera que les fichiers sont bien formés, c'est-à-dire que les lignes ont $3 + n$ champs séparés par des points-virgules, où n est la valeur du troisième champ, et que la somme des valeurs des n derniers champs est égale à la valeur du deuxième champ.

De plus, on supposera dans toutes les questions que les UE ont un nom toujours formé de la même manière, deux lettres suivies de trois chiffres.

Quand une valeur à calculer prend une valeur non entière, on convient de renvoyer son arrondi à l'entier inférieur.

Question 1

On souhaite dans un premier temps savoir comment sont répartis les élèves dans les groupes, c'est-à-dire, UE par UE, combien d'élèves il y a en moyenne par groupe.

Écrire un script `moyenne.sh` qui prend deux arguments : le fichier de descriptif et le code de l'UE, et renvoie la moyenne des effectifs des groupes de l'UE désignée.

Écrire un script `toutesMoyennes.sh` qui prend un argument correspondant au fichier de descriptif, et affiche toutes les moyennes des effectifs des groupes des UE décrites (une ligne par UE, sous la forme `code de l'UE;moyenne`).

On rappelle que cette moyenne s'obtient avec la formule

$$\text{Moyenne} = \frac{\text{Nombre d'étudiants}}{\text{Nombre de groupes}}$$

```
% ./moyenne.sh  LI218 liste1
40
% ./toutesMoyennes.sh liste1
LI218;40
LI215;25
LI315;16
```

Question 2

L'organisation n'a pas forcément été optimale, et on manque de professeurs et de salles. On décide qu'il faut éviter les groupes de moins de 10 élèves, et on cherche à connaître la liste des UE concernées.

Écrire un script `petitsGroupes.sh` qui prend en argument un fichier de descriptif, et affiche la liste des UE qui ont au moins un groupe de moins de 10 élèves (inclus).

```
% ./petitsGroupes.sh liste1
LI215
LI315
```

Question 3 (difficile)

Le résultat du script précédent est loin d'être satisfaisant, il y a beaucoup trop d'UE concernées. On va régler le problème pour cette année, mais on voudrait savoir qui est responsable de cette erreur d'organisation, afin qu'il y remédie l'an prochain. Les groupes d'élèves sont gérées par le responsable de la formation dont dépend l'UE. Pour chaque UE, cette formation correspond aux trois premiers caractères du code de l'UE, c'est-à-dire que LI218 et LI215 appartiennent toutes les deux à la formation LI2, alors que LI315 appartient à la formation LI3.

Écrire un script `moinsRemplie.sh` qui prend en argument un fichier de descriptif, et affiche la formation la moins remplie, c'est-à-dire celle qui a le plus de groupes de moins de 10 élèves.

```
% ./moinsRemplie.sh liste1
LI3
```

Question 4

On veut à présent savoir dans quelles UE on peut améliorer l'organisation. On veut que tous les groupes aient moins de 32 élèves, et avoir le moins de groupes possibles par UE. Plusieurs situations peuvent se produire :

— ou bien l'UE n'a pas assez de groupes : c'est le cas si le nombre moyen d'élèves par groupe est supérieur à 32 ;
— ou bien l'UE a trop de groupes : c'est le cas si $32 * (n - 1) \geq N$ où n est le nombre de groupes et N le nombre d'élèves.

Écrire un script `optimisation.sh` qui prend en argument un fichier de descriptif, et affiche la liste des UE où il y a soit trop de groupes, soit pas assez de groupes, en précisant.

```
% ./optimisation.sh liste1
LI218: pas assez de groupes
LI315: trop de groupes
```

Question 5

On souhaite aller plus loin par rapport au script précédent, et savoir combien de groupes il faut ajouter ou supprimer à chaque UE pour arriver à l'objectif précédent. Écrire un script `optimisationDetaillee.sh` (construit à partir de `optimisation.sh`) qui ajoute cette information.

```
% ./optimisationDetaillee.sh liste1
LI218: pas assez de groupes (2)
LI315: trop de groupes (1)
```

Question 6

On veut à présent vérifier que les fichiers descriptifs sont bien formés, c'est-à-dire qu'ils contiennent autant d'effectifs de groupes que de groupes, et que la somme de ces effectifs vaut bien l'effectif total.

Écrire un script `verification.sh` qui prend comme argument un fichier de descriptif, et affiche en sortie la liste des lignes mal formées, avec un message d'erreur adéquat. On s'assurera que le code de retour du script est positif si au moins une ligne est mal formée, et nul sinon.

Vous trouverez des exemples de fichiers mal formés dans le dossier `/Infos/lmd/2013/licence/ue/li218-2013oct/tmesolo/Groupe4/`, appelés `malforme*`.

```
% ./verification.sh liste1
% echo $?
0
% ./verification.sh malforme1
LI218: 5 groupes annoncés, mais 4 effectifs donnés
% echo $?
1
% ./verification.sh malforme2
LI218: 5 groupes annoncés, mais 6 effectifs donnés
% ./verification.sh malforme3
LI218: 200 élèves annoncés, mais la somme des effectifs des groupes vaut 201
LI315: 3 groupes annoncés, mais 2 effectifs donnés
```