Exercices TP

Recopiez le contenu du répertoire `/Infos/lmd/2011/licence/ue/li219-2012fev/TP8` qui contient tous les fichiers nécessaires pour la réalisation de ce TP et en particulier le fichier `compte_rendu_TP8.txt`que vous devez compléter au fur et à mesure et soumettre à la fin de la séance.

Exercice 53 – Synchronisations multiples

Considérons les trois tâches suivantes :
- saisir des notes : cette tâche consiste à écrire un ensemble de notes dans un fichier,
- calculer une somme de notes : cette tâche consiste à calculer la somme des valeurs stockées dans un fichier et à remplacer le contenu du fichier par le nombre de notes et leur somme,
- calculer une moyenne de notes : cette tâche consiste à calculer la moyenne des notes à partir de leur nombre et de leur somme stockées dans un fichier.

Ces trois tâches sont exécutées par des processus différents. Les scripts exécutés sont `saisie.sh` pour la première tâche, `somme.sh` pour la deuxième et `moy.sh` pour la dernière.

Question 1
Écrivez le script `saisie.sh` qui prend en paramètre un chaîne de caractères. S'il n'y a pas exactement 1 paramètre ou si le paramètre correspond à un répertoire existant, le script se termine.

Le script demande à l'utilisateur de saisir des notes et écrit ces notes dans un fichier dont le nom est la valeur du paramètre (une note par ligne). Le script s'arrête dès qu'une note négative est saisie. Cette note négative est écrite à la fin du fichier.

Question 2
Écrivez le script `somme.sh` qui prend en paramètre un nom de fichier. S'il n'y a pas exactement un paramètre ou si le paramètre ne correspond pas à un fichier, le script se termine. Le script calcule la somme des notes positives ou nulles se trouvant dans le fichier et modifie son contenu en le remplaçant par le nombre de notes et leur somme. le fichier doit initialement contenir une note positive par ligne et se terminer par une note négative (comme le fichier produit par le script `saisie.sh`).

Question 3
Écrivez le script `moy.sh` qui prend en paramètre un nom de fichier. S'il n'y a pas exactement un paramètre ou si le paramètre ne correspond pas à un fichier, le script se termine. Le fichier doit contenir deux entiers, un par ligne, le premier correspond à un nombre de notes et le second à une somme de notes (comme le fichier produit par le script `somme.sh`). Le script calcule la moyenne entière correspondant aux informations contenues dans le fichier. Si cette moyenne est supérieure ou égale à 10, il affiche `recu` suivi de la moyenne sinon, il affiche `recale` suivi de la moyenne. Ce script ne modifie pas le contenu du fichier.

Question 4
Que se passe-t-il lors de l'exécution suivante ?

```
Prompt% cat fic_notes
7
12
9
8
15
3
-4
Prompt% ./saisie.sh fic < fic_notes & ./somme.sh fic & ./moy.sh fic &
```

Question 5
Écrivez une nouvelle version des scripts qui garantit que les problèmes identifiés à la question précédente ne se présentent pas.

Question 6
Si, à la fin de l'exécution suivante, le fichier `res` ne contient pas les valeurs 6 et 54 et le fichier `res1` les valeurs 3 et 21, c'est probablement que les modifications que vous avez apportées à la question précédente synchronisent les processus sans se soucier du fichier dans lequel ils écrivent ou lisent. Dans ce cas, modifiez vos scripts pour que les résultats obtenus soient corrects.

```
Prompt% cat fic_notes
7
12
9
8
15
3
-4
Prompt% cat fic_notes1
10
5
6
-2
Prompt% ./saisie.sh res < fic_notes & ./somme.sh res & ./moy.sh res & ./saisie.sh
res1 < fic_notes1 & ./somme.sh res1 & ./moy.sh res1 &
```

Exercice 54 – Synchronisations croisées et alternance d'exécutions

Question 1
Écrivez un script `ecrivain.sh` qui prend au moins 2 paramètres. Le premier paramètre est une chaîne de caractères. Si la valeur de ce paramètre correspond à un répertoire existant, le script se termine avec un message d'erreur. Sinon, le script écrit successivement les valeurs des autres paramètres dans un fichier dont le nom correspond au premier paramètre. La suite de commandes

```
Prompt% ls
 ecrivain.sh     lecteur.sh
Prompt% ./ecrivain.sh fic_test 3 2 6 4
```

a donc pour effet de créer un fichier `fic_test` qui contient les valeurs 3, 2, 6, 4 (une par ligne) à la fin de l'exécution. Si le fichier existait déjà son contenu est remplacé.

Question 2
Écrivez un script `lecteur.sh` qui prend exactement 1 paramètre. Si ce paramètre ne correspond pas à un fichier, le script termine son exécution avec un message d'erreur. Sinon, il lit le contenu du fichier ligne par ligne. Pour chaque mot lu, il affiche "`mot lu :` ", suivi de la valeur récupérée.

Nous souhaitons maintenant synchroniser le lecteur et l'écrivain de manière à ce que chaque valeur écrite dans le fichier soit ensuite lue, mais sans imposer une exécution séquentielle du lecteur et de l'écrivain. Pour mettre en évidence les problèmes qui peuvent se produire, nous ajoutons une instruction `sleep 1` dans la boucle d'écriture.

Question 3
Exécutez plusieurs fois la commande suivante (en n'oubliant pas de détruire le fichier `fic_test` entre deux exécutions). Que constatez-vous ?

```
Prompt% ./ecrivain.sh fic_test 1 2 3 4 5 & ./lecteur.sh fic_test &
```

Nous souhaitons maintenant que le lecteur lise (et donc affiche) chacune des valeurs écrites par l'écrivain. Pour obtenir une solution qui fonctionne aussi bien dans le cas où l'écrivain ajoute des valeurs dans le fichier, que dans le cas où chaque écriture écrase la valeur précédente, nous allons forcer une alternance stricte entre les écritures et les lectures. Il faut donc garantir que :

– la première opération effectuée est une écriture ;
– l'écrivain attend pour faire une nouvelle écriture que son écriture précédente ait été lue ;
– le lecteur ne fait pas de nouvelle lecture s'il n'y a pas eu de nouvelle écriture.

Lorsqu'il a terminé ses écritures, l'écrivain écrit la chaîne "fin" dans le fichier. Lorsqu'il lit cette valeur, le lecteur sait donc qu'il peut se terminer.

Question 4

Modifiez les deux scripts pour qu'ils se synchronisent correctement. Vérifiez la synchronisation dans le cas où le fichier contient toutes les valeurs écrites et dans le cas où il ne contient jamais plus d'une valeur (chaque écriture écrase la précédente).