INF203 - Travaux pratiques, séance 2

Redirections des entrées-sorties, shell-script

[INF203] Lors de la dernière séance de TP vous avez écrit un fichier de commande de nom `installeTP.sh`. Relisez-le et utilisez-le pour créer un répertoire ***TP2*** et y copier les fichiers nécessaires à cette séance.

Redirection des entrées et/ou des sorties d'un programme

[TP2] Sous Unix, les *sorties* produites à l'écran par l'exécution d'un programme (et donc d'une commande Unix) peuvent être **redirigées vers** un fichier en utilisant le symbole ">". Exécutez la séquence de commandes suivante :

```
date
date > une_date
cat une_date
```

Remarque Exécutez la commande :

```
ls > une_date
```

Le fichier ***une_date*** qui existait déjà est écrasé "silencieusement" (il est remplacé par un nouveau fichier, et l'ancien est perdu) : vérifiez-le.

De la même manière, les *entrées* lues au clavier par un programme (ou une commande Unix) peuvent être **redirigées depuis** un fichier en utilisant l'opérateur "<". Lisez le contenu du fichier ***max2.c***. Compilez ce programme source en un exécutable de nom ***max2*** (commande `gcc max2.c -o max2`).
[a] Notez la taille du fichier ***max2***. Exécutez le programme ***max2*** en notant soigneusement sur votre compte rendu ce qui s'affiche à l'écran, en distinguant à l'aide de deux couleurs ce qui correspond aux sorties du programme, et ce qui correspond à l'écho de ce qui a été saisi au clavier. ■

Créez (avec **nedit**) un fichier ***deux_entiers*** contenant uniquement deux entiers sur une ligne (séparés par un ou plusieurs espaces). Essayez la commande suivante pour exécuter ***max2*** en redirigeant les entrées depuis le fichier ***deux_entiers*** :

```
./max2 < deux_entiers
```

[b] Notez les messages d'erreur que vous obtenez si vous exécutez : `./deux_entiers > ./max2`.
Donnez-vous les droits en exécution sur le fichier **deux_entiers** et ré-essayez la commande précédente. Quel est alors le message d'erreur obtenu ? Quelle est désormais la taille du fichier ***max2*** et que contient-il ? Pourquoi ? ■

Recompilez ***max2.c***.
Essayons maintenant de rediriger les sorties de ***max2*** vers un fichier ***resultat*** :

```
./max2 > resultat
```

[c] Pourquoi ne se passe-t-il "rien" ? ■
Quand vous aurez trouvé la réponse à cette question, et réagi en conséquence, vérifiez que le fichier ***resultat*** contient bien le résultat attendu.

Enfin, il est également possible de rediriger à la fois les entrées et les sorties d'un programme. Essayez par exemple :

```
./max2 < deux_entiers > resultat1
./max2 > resultat2 < deux_entiers
```

Les fichiers ***resultat1*** et ***resultat2*** sont-ils identiques ?

Enchaînement de l'exécution de deux programmes.
1) date et instant_suivant

[TP2] Lisez le fichier ***instant_suivant.c***. Il contient un programme incomplet dont le comportement doit être le suivant : étant donné un instant (saisi au clavier), exprimé sous la forme HH:MM:SS (par exemple 09 :30 :25), le programme affiche l'instant à la seconde qui suit (par exemple 09 :30 :26).
Observez la définition du type `instant` : un `instant` est composé de 3 entiers appelés `heure`, `minute` et `seconde`. On accède à ces entiers (les *champs* du type) de la manière suivante : si `x` est une variable de type `instant`, alors `x.heure` désigne le champ `heure` de `x`, ... Complétéz, compilez et testez ***instant_suivant***
[d] Combien de tests (et quels tests) faut-il faire pour vérifier tous les cas de figure ? ■

La commande ***date*** utilisée avec l'option `+%T` permet d'afficher l'heure au format $HH:MM:SS$. On veut maintenant calculer l'instant suivant de celui donné par la commande **date**. Exécutez les commandes :

```
date +%T
date +%T > une_date
./instant_suivant < une_date
```

On peut obtenir le même résultat sans créer le fichier ***une_date***, en enchaînant directement l'exécution des programmes ***date*** et ***instant_suivant***, en une seule commande. On utilise pour cela une possibilité offerte par Unix, le *pipeline*. Pour "pipeliner" l'exécution de deux commandes, on utilise l'opérateur "pipe" (prononcez païpe), noté |. Exécutez plusieurs fois la commande :

```
date +%T | ./instant_suivant
```

[e] Joignez le texte de ***instant_suivant.c*** à votre compte rendu. ■

Exercice complémentaire :

Lisez le manuel de la commande **date** (`man date`) et essayez différentes options d'affichage. Utilisez la commande **date** pour afficher la date de dernière modification de ***instant_suivant.c***.
[f] Quelles options de `date` utilisez-vous pour afficher la date de dernière modification de ***instant_suivant.c*** ?
■

2) ls et less
L'opérateur "pipe" est très fréquemment utilisé avec la commande **less** lorsque les résultats affichés à l'écran par un programme (ou une commande Unix) dépassent largement le nombre de lignes disponibles dans la fenêtre. Exécutez la commande `ls -lRa ~` qui affiche la liste de tous les fichiers et répertoires de votre arborescence. Pour pouvoir vraiment lire ce qui est affiché, exécutez maintenant cette commande en la "pipelinant" avec la commande **less**. **[g]** Quelle est la commande que vous saisissez pour pipeliner `ls -lRa ~` dans **less** ? ■

Structures de contrôle en shell

Instructions conditionnelles.

1) Exemple

[TP2] Lisez le texte du programme ***max2.sh***, qui affiche le maximum de 2 entiers donnés en arguments.
[h] Quels tests effectuez-vous pour vérifier que ce programme est "correct" dans tous les cas de figure possibles ?
■

Exercice complémentaire :

[i] Dans le programme ***max2err.sh***, une erreur a été commise : laquelle ? Cependant, ce programme fonctionne bien pour certaines configurations des données, lesquelles ? En quoi ceci a-t-il un rapport avec la terminologie que nous utilisons : "interpréteur de commandes" ?
Si une erreur du même genre est commise dans un programme écrit en C, ce programme pourrait-il s'exécuter dans certains cas seulement ? ■

2) Exécutez la commande ***max2.sh*** avec un nombre insuffisant d'arguments. Modifiez ***max2.sh*** pour afficher un message d'erreur dans ce cas de figure (indication : utilisez `$#` pour tester que le nombre d'arguments fournis est correct).

[j] Joignez le texte de ***max2.sh*** à votre compte-rendu. ■

Exercice complémentaire :

[INF203] Modifiez le fichier de commande ***installeTP.sh*** de manière à ce que la commande `installeTP.sh i`
- s'exécute normalement lorsque le répertoire ***TPi*** n'existe pas dans le répertoire courant
- affiche un message d'erreur dans le cas contraire.

On rappelle que la condition "`[ -d R ]`" vaut vrai si et seulement si `R` est un nom de répertoire présent dans le répertoire courant. Testez cette nouvelle version de ***installeTP.sh***.

Instructions d'itération : boucle for

1) Exemple

[TP2] Lisez le texte du programme ***description.sh***. Il indique, pour chaque élément du répertoire courant, s'il s'agit d'un fichier exécutable ou non, ou d'un répertiore.

Exercice complémentaire :

Modifiez ce programme pour qu'il crée un répertoire ***Exec*** dans le répertoire courant et y copie les fichiers exécutables.
[k] Testez cette nouvelle version et joignez le texte à votre compte-rendu. ■

2) Le programme ***des_ dates.sh*** affiche 10 fois la date, à intervalles de 2 secondes. Lisez-le et exécutez-le. Créez une copie de ***instant_ suivant.c*** nommée ***instant_ suivant10.c*** et modifiez ce programme pour qu'il répète 10 fois les opérations de lecture d'une date au clavier et d'affichage de l'instant suivant cette date. Testez-le en lui fournissant en entrée les sorties produites par ***des_ dates.sh***.
[l] Quelle(s) commande(s) utilisez-vous pour cela ? ■

Exercice complémentaire :

Modifiez maintenant le programme ***des_ dates.sh*** pour que le nombre d'itérations et le temps de pause entre deux affichages soient fournis en arguments de la ligne de commande.
[m] Testez cette nouvelle version et joignez le texte à votre compte rendu. ■
Essayez maintenant d'utiliser les sorties produites par cette nouvelle version de ***des_ dates.sh*** comme entrées de ***instant_ suivant10*** avec pour valeur du nombre d'itérations, 10 puis 5 puis 15, par exemple..
[n] Quels comportements et/ou messages d'erreurs constatez-vous ? ■

Des choses utiles à connaître

La commande de la semaine : `diff`

Pour savoir si les contenus de 2 fichiers sont identiques, on peut utiliser la commande **diff**. Elle donne 3 types de réponses :
- aucune réponse : les 2 fichiers sont identiques.
- la réponse est seulement que les 2 fichiers sont différents : l'un au moins n'est pas un fichier texte.
- la liste des différences entre les 2 fichiers : les 2 fichiers sont des fichiers texte.

Essayez d'obtenir ces 3 types de réponses avec la commande **diff**.

Code de retour des commandes

Lorsqu'il y a un problème lors de l'exécution d'une commande unix, cela provoque l'affichage d'un message d'erreur. Exécutez successivement les commandes suivantes, et notez les messages qui sont affichés, et à quel moment :

```
mkdir DIR
mkdir DIR
rmdir DIR
cd DIR
mkdir DIR
cd DIR
```

```
cp ../copiedir.sh cop
cp ../../copiedir.sh cop
cp cop
```

Outre l'affichage ou non d'un message, les commandes "retournent" un code entier, qui est différent selon qu'il y a eu ou non un problème lors de l'exécution. On peut connaître ce code en affichant la variable `$?` juste après l'exécution de la commande.
Supprimez tous les fichiers du répertoire ***DIR*** et le répertoire ***DIR*** lui-même. Refaites la suite de commandes ci-dessus, et après chacune d'elles, exécutez la commande

```
echo $?
```

[o] A quoi correspondent les codes de retour obtenus (erreur/pas erreur) ? ■

Enlever le suffixe d'un nom

Lisez et exécutez ***prefixe.sh***, et vérifiez que vous comprenez son fonctionnement.

Exercice de synthèse

Il s'agit d'écrire un script qui vérifie qu'un programme fournit les résultats attendus. Le programme à tester est ***instant_suivant.c*** que vous avez corrigé en début de TP.
Créez un répertoire ***TEST_INSTANT_SUIVANT***, placez-vous dans ce répertoire et copiez-y ***instant_suivant.c***.
[TEST_INSTANT_SUIVANT] Compilez votre programme, et révisez son fonctionnement, en particulier quelles sont ses entrées et ses sorties. Modifiez-le de façon à ce que les entrées et les sorties soient réduites à :

- en entrée : une heure au format HH:MM:SS
- en sortie : l'instant à la seconde qui suit, au même format ; supprimez donc les messages superflus comme `nouvelle heure`, par exemple.

Votre programme doit avoir le comportement suivant :

```
./instant_suivant
14:47:25
14:47:26
```

Créez un certain nombre de fichiers d'entrée pour ***instant_suivant*** : dans cette version, chaque fichier contient exactement une heure au format HH:MM:SS, et suffixez ces fichiers par `.entree`. Par exemple, le fichier `144725.entree` contient la chaîne `14:47:25`.
Pour chacun de ces fichiers, créez un fichier de même préfixe mais suffixé par `.sortie` contenant le résultat attendu du programme. Par exemple, le fichier `144725.sortie` contient `14:47:26`.

1) Test élémentaire Éxécutez les commandes :

```
./instant_suivant < 144725.entree > 144725.ma_sortie
diff 144725.sortie 144725.ma_sortie
echo $?
```

Est-ce que le test est réussi ?
Déplacez le fichier ***test_elem.sh*** du répertoire ***[TP2]*** dans ***[TEST_INSTANT_SUIVANT]*** (ici) :

```
mv ../test_elem.sh .
```

Lisez, comprenez et exécutez ce programme.

2) Et maintenant, écrivez un script `test_instant.sh` qui vérifie que votre programme ***instant_suivant*** fournit bien les résultats attendus pour chacun des fichiers `.entree`. Pour cela, vous devez réutiliser un certain nombre d'éléments vus au cours de ce TP comme :

- le parcours dans une boucle `for` de tous les fichiers du répertoire suffixés par `.entree`
- l'extraction du préfixe de chacun des noms de fichiers
- ...

Testez ce programme, et utilisez-le pour vérifier que votre programme ***instant_suivant*** est correct.
[p] Joignez le texte de ***test_instant.sh*** à votre compte rendu. ■

En fin de séance ...

[INF203] A la fin de chaque séance vous devez systématiquement copier l'ensemble du travail effectué sur le compte de votre binôme. Nous vous conseillons pour cela d'utiliser la commande ***scp*** ("secure copy") qui permet l'accès à des répertoires d'autres utilisateurs via la saisie de leur mot de passe. Ainsi, pour copier votre répertoire TP2 et son contenu dans le répertoire INF203 de votre binôme, vous pouvez utiliser la commande suivante :

```
scp -r TP2 login_binome@turing.e.ujf-grenoble.fr:INF203
```