TP2 : Interpréteur de commandes, shell, redirection

CSH : Initiation au C et au shell

Première année

1 Shell

*[**TP1**]* Relisez le contenu du fichier ***copiedir.sh***.
On souhaite généraliser ce fichier de commande (on l'appellera ***installeTP.sh***). On l'utilisera au début de chaque séance de TP en lui indiquant en argument le «numéro» de la séance (un entier). Ainsi, par exemple, la commande `installeTP.sh 4` permettra de créer un répertoire ***TP4*** dans le répertoire principal, et de recopier dans ce répertoire le contenu du répertoire ***/home/depot/1A/CSH/TP4***.

Faites une copie du fichier ***copiedir.sh***, dans le répertoire principal ***CSH***, sous le nom ***installeTP.sh***. Modifiez ce fichier pour qu'il corresponde à ce qui est présenté ci-dessus.

*[**CSH**]* Assurez-vous que votre travail est correct en exécutant la commande `installeTP.sh 2` et en vérifiant qu'un répertoire ***TP2*** a bien été créé dans le répertoire ***CSH***, et que ce répertoire ***TP2*** n'est pas vide.

Quel message d'erreur obtenez-vous lorsque vous exécutez cette commande plusieurs fois de suite ? Que signifie-t-il ? Le contenu du répertoire ***TP2*** est-il modifié par l'exécution de cette commande ?

Si le répertoire ***CSH*** n'existait pas (ou ne se trouvait pas au bon endroit dans l'arborescence des répertoires), quel message d'erreur provoquerait cette exécution de `./installeTP.sh 1` ? (recherchez dans la liste établie lors de la première séance)

Exécutez la commande `./installeTP.sh` en «oubliant» de fournir un argument. Que se passe-t-il ? Réparez les dégâts.

2 Re-direction 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 l'opérateur “>”. Essayez par exemple les commandes suivantes :

```
1 date
2 date > une_date
3 less une_date
```

Remarque sur l'opérateur “>” : si vous exécutez à nouveau la commande :

```
1  date > une_date
```

vous obtenez un message d'erreur, car le fichier ***une_date*** existe déjà. Si vous voulez rediriger de nouveau les sorties vers un fichier qui s'appelle ***une_date***, vous pouvez, soit détruire d'abord le fichier existant, soit exécuter directement la commande :

```
1 date > ! une_date
```

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*** qui se trouve dans votre répertoire.

Compilez ce programme source en un exécutable de nom ***max2*** (commande `gcc max2.c -o max2`). Exécutez-le 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 `emacs`) un fichier ***deux_entiers*** contenant uniquement deux entiers sur la première 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
```

Notez les messages d'erreur que vous obtenez si vous exécutez :

```
./deux_entiers > max2
./deux_entiers
```

Essayons maintenant de rediriger les sorties de `max2` vers un fichier ***resultat*** :

```
./max2 > resultat
```

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 ?

Pour savoir si les contenus de deux fichiers sont identiques, on peut utiliser la commande `diff`. Elle donne trois 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 chacune de ces réponses avec la commande `diff`.

3 Un peu de C pour changer

[TP2] Lisez le texte du fichier ***instant_suivant.c***, et complétez les parties indiquées afin que le programme affiche (sous la forme HH :MM :SS) l'instant à la seconde qui suit l'instant saisi au clavier. Compilez, et testez votre programme.

On veut maintenant calculer l'instant suivant de celui donné par la commande `date`. Pour cela, dans le programme ***instant_suivant.c***, remplacez l'instruction `scanf` par celle-ci :

```
scanf("%s%s%d%2d:%2d:%2d", jour, mois, &quant, &heure, &min, &sec);
```

Ajoutez aussi les déclarations correspondant à cette modification : `jour` et `mois` sont des tableaux de quatre caractères, et `quant` est un entier. Compilez.

Exécutez les commandes :

```
date > une_date
./instant_suivant < une_date
```

Comment vérifiez-vous que ce qui est affiché est correct ?

4 Enchaînement de l'exécution de deux programmes

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 *tube* («pipe» en anglais). Pour «tuber» l'exécution de deux commandes, on utilise l'opérateur «tube» (ou «pipe», à prononcer païpe), noté |. Exécutez plusieurs fois la commande :

```
date | instant_suivant
```

L'opérateur «tube» est très fréquemment utilisé avec la commande `less` (ou `more`) 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 :

```
1 less TP-02.tex
```

Le fichier ***TP-02.tex*** contient le texte de cet énoncé, en LaTeX. Grâce à la commande `less`, vous en voyez la première page d'écran. Avec la touche <espace>, vous passez à la page suivante, avec la touche <entrée>, vous avancez d'une ligne seulement. Vous pouvez quitter le programme avec la touche <q>.

Placez-vous dans votre répertoire personnel et 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 «tubant» dans la commande `less`.

5 Compilation et exécution d'un programme C

[TP2] Écrivez un programme C qui calcule le maximum de 3 entiers saisis au clavier par l'utilisateur, et qui affiche le résultat. Compilez-le sous le nom ***max3***, et vérifiez que votre programme est correct. Quels tests effectuez-vous pour cela ?

6 Exercices pour la prochaine fois

1. Écrivez un programme C qui permette de lire au clavier une suite de caractères représentant la plaque minéralogique d'un véhicule (comme `982 BZZ 54`) et qui affiche à l'écran les plaques des 20 véhicules immatriculés à la suite de celui-ci (dans le même département).

   Ainsi, si la chaîne lue est «`998 BZZ 54`», alors le programme affichera
   «`999 BZZ 54`», «`001 CAA 54`», «`002 CAA 54`», ..., «`019 CAA 54`».
2. On suppose que ce programme est compilé en un exécutable de nom ***plaques_suivantes***. Quelle(s) commande(s) Unix faut-il utiliser pour éxécuter ce programme si l'on suppose que la chaîne fournie en entrée se trouve dans un fichier ***plaque*** et que l'on souhaite écrire le résultat dans un fichier de nom ***resultat_plaques*** ?