Devoir surveillé du 23 mai 2006

CSH : Initiation au C et au shell

Première année

Tous documents interdits. Les exercices sont indépendants. La correction tiendra compte de la qualité de la rédaction et de la présentation. Barème approximatif.

★ **Exercice 1. (4pts)** L'utilisateur `laurel` examine le contenu du répertoire courant :

```
{laurel} 1 > ls -l
total 92
-rw-r--r--   1 laurel esial1     226 avr  4 10:08 Makefile
drwxr-xr-x   2 laurel esial1     512 avr  7 07:03 TP1
drwxr-xr-x   2 laurel esial1     512 avr  7 07:03 TP2
-rw-r--r--   1 laurel esial1     129 mar 10 17:12 deux.c
-rw-r--r--   1 laurel esial1      12 jun 11 15:04 deux_entiers
-rw-r--r--   1 laurel esial1     225 jun 10 17:13 trois.c
-rw-r--r--   1 laurel esial1     173 mar  7 07:07 installeTP.sh
-rwxr-xr-x   1 laurel esial1    1115 jun  8 17:04 max2
-rw-r--r--   1 laurel esial1     897 mai 10 17:14 un.c
-rw-r--r--   1 laurel esial1     154 mai 27 15:29 y.c
```

▷ **Question 1. (2pts)** Laurel exécute alors différentes commandes qui se terminent toutes par un message d'erreur. Expliquez dans chaque cas le sens de ces messages et indiquez comment remédier au problème.

a
```
{laurel} 2 > more deux_entiers
23 12
{laurel} 3 > max2 < resultat > deux_entiers
resultat: No such file or directory.
{laurel} 4 >
```

b
```
{laurel} 5 > which installeTP.sh
installeTP.sh: Command not found.
{laurel} 6 >
```

c
```
{laurel} 7 > gcc -o y y.c
Undefined                       first referenced
 symbol                             in file
ploumploum                          /var/tmp//ccMi6qE9.o
ld: fatal: Symbol referencing errors. No output written to y
collect2: ld returned 1 exit status
{laurel} 8 >
```

d
```
{laurel} 9 > touch *.c
{laurel} 10 > make
gcc -c un.c
gcc -c deux.c
gcc -c trois.c
gcc -o quatre un.o deux.o trois.o
ld: fatal: symbol `zero' is multiply-defined:
        (file deux.o type=FUNC; file trois.o type=FUNC);
ld: fatal: File processing errors. No output written to quatre
collect2: ld returned 1 exit status
*** Error code 1
make: Fatal error: Command failed for target `quatre'
{laurel} 11 >
```

▷ **Question 2. (1pt)** Écrivez un contenu possible pour le fichier ***Makefile***.

▷ **Question 3. (1pt)** L'utilisateur `hardy` a aussi écrit un ***Makefile*** dans son répertoire personnel. Comment Laurel sait-t-il s'il peut lire le ***Makefile*** de son ami ? Quelle(s) commande(s) pourra-t-il alors utiliser pour comparer ces deux fichiers ***Makefile*** ?

★ **Exercice 2. (3pts)** La commande ***strings*** permet d'étudier les parties lisibles d'un fichier binaire dont le nom est passé en argument. Elle n'affiche à l'écran que les caractères du fichier dont les codes Ascii sont dans l'intervalle $[32, 126]$ (ce sont les caractères "affichables", les autres étant des des codes de contrôle).

Exemple : si ***fichier*** contient les caractères de codes Ascii $(12, 33, 65, 69, 5, 4, 77, 89, 14, 35, 101, 102)$ alors `strings fichier` affichera les caractères : $(33, 65, 69, 77, 89, 35, 101, 102)$. Soit : `!AEMY#ef`

▷ **Question 1.** Écrivez en C la commande ***strings***.

★ **Exercice 3. (3pts)** Écrivez un programme C ***min_maj.c***, compilé sous le nom ***min_maj***, qui affiche à l'écran ses arguments en convertissant toutes les lettres minuscules en lettres majuscules (les autres caractères étant inchangés).
Exemples :
- `min_maj bOnNE nUIt LES petits ...` affiche à l'écran : `BONNE NUIT LES PETITS ...`
- `min_maj Le C eT Le ShEll, tROp 3ll3t` affiche à l'écran : `LE C ET LE SHELL, TROP 3LL3T`

Indication : le décallage entre les majuscules et les minuscules est donné par l'expression `'A'-'a'`.

★ **Exercice 4. (2pts)** Écrivez un fichier de commandes de nom ***sauvegarde.sh*** permettant de :
- Créer un répertoire de nom ***Archives*** dans votre répertoire personnel (s'il n'y existe pas déjà) ;
- Déplacez dans ce répertoire ***Archives*** tous les fichiers exécutables qui se trouvent dans le répertoire dont le nom est donné en argument.
  Si le répertoire donné en argument n'existe pas, afficher un message d'erreur.

★ **Exercice 5. (5pts)** La commande `wc -l` affiche le nombre de lignes du fichier donné en arguments :

```
{batman} 2 > wc -l debordechar.c
      35 debordechar.c
{batman} 3 >
```

▷ **Question 1. (2pts)** Écrivez un programme C `compte_ligne` mimant l'action de `wc -l`. Il prend un nom de fichier en argument et affiche le nombre de lignes du fichier suivi du nom du fichier.

▷ **Question 2. (subsidiaire, 0pt)** Écrivez un script shell `compte_ligne.sh` équivalent.
Indication : `grep -c` affiche le nombre d'occurences du motif passé en argument.

On souhaite maintenant connaître le nombre total de lignes de tous les fichiers suffixés par `.c` ou `.h` du répertoire courant.

▷ **Question 3. (1pt)** Décrivez en quelques lignes le principe d'une solution possible. Cette solution peut combiner l'utilisation de commandes existantes ainsi que des programmes C ou fichiers de commandes (dont vous préciserez les fonctionnalités). Vous pouvez utiliser `compte_ligne`, mais pas `wc`.

▷ **Question 4. (2pts)** Donnez la réalisation complète de votre solution.

★ **Exercice 6. (3pts)** On suppose que l'on dispose d'un fichier calepin.txt, contenant des noms et des numéros de téléphone rangés de la manière suivante :

calepin.txt
```
DUPONT Jean,05.61.75.18.47,21/08/1975,jean.dupont@free.fr
MARTIN Yvonne,02.23.34.45.56,26/02/1977,yvonne.martin@cegetel.fr
...
```

▷ **Question 1. ($\frac{1}{2}$pt)** Écrire un script (ou une commande) qui effectue la recherche des personnes s'appelant DURAND et qui restitue leur numéro de téléphone.

▷ **Question 2. ($\frac{1}{2}$pt)** Écrire un script (ou une commande) qui compte les gens habitant dans le sud-est (numéro en 04)

▷ **Question 3. (1pt)** Écrire un script (ou une commande) envoyant un message éléctronique contenant "Bonne Année" à tout le monde.

▷ **Question 4. (1pt)** Écrire un script (ou une commande) envoyant un message éléctronique contenant "Bon anniversaire" à toutes les personnes dont c'est l'anniversaire.
Indication : `date +%d/%m` affiche la date du jour sous la forme : jj/mm