UniversitÃ© Joseph Fourier **UE INF122**
Licence L1 "Sciences et Technologies"

Devoir SurveillÃ© du 14 janvier 2005

DurÃ©e 1h30 ; Documents autorisÃ©s
Les exercices sont indÃ©pendants

Exercice 1

L'utilisateur `laurel` dÃ©bute une session de travail en examinant le contenu du rÃ©pertoire courant :

```
{laurel} 5 >ls -l
total 92
-rw-r--r--   1 laurel dsu-etu     226 Jan  4 10:08 Makefile
drwxr-xr-x   2 laurel dsu-etu     512 Jan  7 07:03 TP1
drwxr-xr-x   2 laurel dsu-etu     512 Jan  7 07:03 TP2
-rw-r--r--   1 laurel dsu-etu     129 Nov 10 17:12 deux.c
-rw-r--r--   1 laurel dsu-etu      12 Nov 11 15:04 deux_entiers
-rw-r--r--   1 laurel dsu-etu     225 Nov 10 17:13 trois.c
-rw-r--r--   1 laurel dsu-etu     173 Jan  7 07:07 installeTP.sh
-rwxr-xr-x   1 laurel dsu-etu    1115 Jan  8 17:04 max2
-rw-r--r--   1 laurel dsu-etu     897 Nov 10 17:14 un.c
-rw-r--r--   1 laurel dsu-etu     154 Dec 27 15:29 y.c
```

1. Laurel exÃ©cute alors diffÃ©rentes commandes qui se terminent toutes par un message d'erreur. Expliquez dans chaque cas le sens de ces messages d'erreurs et indiquez comment remÃ©dier au problÃ¨me.

1. ```
   {laurel} 2 >more deux_entiers
   23 12
   {laurel} 3 >max2 < resultat > deux_entiers
   resultat: No such file or directory.
   {laurel} 4 >
   ```

2. ```
   {laurel} 6 >which installeTP.sh
   installeTP.sh: Command not found.
   {laurel} 7 >
   ```

3. ```
   {laurel} 21 >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} 22 >
```

4.
```
{laurel} 55 >touch *.c
{laurel} 56 >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} 57 >
```

2. Ecrivez un contenu possible pour le fichier ***Makefile***.

3. L'utilisateur `hardy` a aussi Ã©crit un ***Makefile*** dans son rÃ©pertoire 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

La commande Unix ***strings*** est une commande qui s'applique Ã un fichier dont le nom est prÃ©cisÃ© en argument de la ligne de commande :

```
strings fichier
```

Cette commande permet d'afficher Ã l'Ã©cran tous les caractÃ¨res contenus dans ***fichier*** dont les codes Ascii sont situÃ©s dans l'intervalle $[32, 126]$ (ce sont les caractÃ¨res "affichables" du code Ascii).

Exemple : si ***fichier*** est constituÃ© des caractÃ¨res dont les codes Ascii sont

$$12, 32, 65, 69, 5, 4, 77, 89, 14, 35, 101, 102$$

alors `strings fichier` affichera Ã l'Ã©cran les caractÃ¨res dont les codes Ascii sont

$$32, 65, 69, 77, 89, 35, 101, 102$$

Ecrivez en C la commande ***strings***.

Exercice 3

Ecrivez un programme C ***min_maj.c***, compilÃ© sous le nom ***min_maj***, qui affiche Ã l'Ã©cran ses arguments en convertissant toutes les lettres minuscules en lettres majuscules (les autres caractÃ¨res Ã©tant inchangÃ©s).

Exemples :

```
min_maj bOnNE nUIt LES petits ...
```

affiche Ã l'Ã©cran :

```
BONNE NUIT LES PETITS ...
```

```
min_maj Vive Inf_122 !
```

affiche Ã l'Ã©cran :

```
VIVE INF_122 !
```

Excercice 4

Ecrivez un fichier de commandes de nom ***sauvegarde.sh*** permettant de :

- CrÃ©er un rÃ©pertoire de nom ***Archives*** dans votre rÃ©pertoire personnel (s'il n'y existe pas dÃ©jÃ) ;
- DÃ©placez dans ce rÃ©pertoire ***Archives*** tous les fichiers exÃ©cutables qui se trouvent dans le rÃ©pertoire dont le nom est donnÃ© en argument. Si le rÃ©pertoire donnÃ© en argument n'existe pas, afficher un message d'erreur.

Exercice 5

L'administrateur d'une machine souhaite Ãªtre prÃ©venu lorsque le nombre d'utilisateurs d'une machine dÃ©passe une certaine valeur N. Il souhaite donc Ã©crire un programme qui lui affiche un message chaque fois que cette valeur est dÃ©passÃ©e.

Pour connaitre le nombre courant d'utilisateurs il dispose de la commande ***uptime*** (vue en TP) :

```
{administrateur} 27 >uptime
  9:16am  up 99 day(s), 23:08,  2 users,  load average: 1.47, 1.60, 1.52
{administrateur} 28 >
```

1. Ecrivez un fichier de commande ***uptime.sh*** qui exÃ©cute la commande ***uptime*** toutes les 10 minutes

2. Ecrivez un programme C qui sera compilÃ© sous le nom ***uptime_vers_users***. Ce programme prend en argument sur la ligne de commande un entier N. Il lit en permanence au clavier le rÃ©sultat de la commande ***uptime*** et affiche un message lorsque le nombre d'utilisateurs dÃ©passe N.

3. Quelle(s) commande(s) doit exÃ©cuter notre administrateur pour qu'un message soit affichÃ© chaque fois qu'il y a plus de 15 utilisateurs sur la machine ?

Exercice 6

La commande `wc -l` affiche Ã l'Ã©cran le nombre de lignes du fichier donnÃ© en arguments :

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

```
      35 debordechar.c
{batman} 3 >
```

On souhaite connaÃ®tre le nombre total de lignes de tous les fichiers suffixÃ©s par `.c` du rÃ©pertoire courant.

1. DÃ©crivez 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 prÃ©ciserez les fonctionnalitÃ©s).

2. Donnez la rÃ©alisation complÃ¨te de votre solution.