IN405 – Système d'exploitation
TD 1 – Terminal et script Shell

S. Gougeaud

2017/2018

L'archive `code-td1.tar` *contient un fichier :*

- `code-mystere.c` – *fichier principal de l'exercice 5, à déboguer.*

Exercice 1 – Compréhension des commandes de base

Soit la liste de commandes suivante : `cat cd cp diff echo gcc gdb ls make man mkdir mv rm rmdir sudo tar time touch vi`

1. Donnez une brève description pour chacune des commandes.
2. Quelles commandes consistent en l'exécution d'un binaire ?
3. Quels chemins sont représentés par les symboles suivants : `.`, `..`, `~`.

Exercice 2 – Première utilisation du terminal

Pour chacune des questions suivantes, exécutez la commande correspondante.

1. Déplacez vous dans le répertoire temporaire de votre système de fichiers.
2. Créez le répertoire `project` ainsi que les sous-répertoires `doc`, `include` et `src`.
3. Au sein du dossier `project`, créez un fichier `README` contenant votre nom et prénom. Créez le fichier `func.h` dans `include`, les fichiers `main.c` et `func.c` dans `src`.
4. Affichez la hiérarchie complète du répertoire `project` et des ses sous-répertoires, puis écrivez ce résultat dans `contents.txt`.
5. Créez une copie du répertoire `project` que vous nommerez `projectV2`. Supprimez le répertoire `project`.
6. Créez l'archive `pv2.tar` contenant l'ensemble du répertoire `projectV2`.

Exercice 3 – Premier script Shell

Afin d'automatiser l'exécution de commandes (comme par exemple la compilation d'un projet ou l'exécution d'un jeu de tests), il est possible de les rassembler dans un fichier. Ce type de fichier est appelé script. Placez l'ensemble des commandes écrites dans l'exercice 2 dans un script Shell, et exécutez-le. Le résultat est-il le même que dans l'exercice 2 ?

Exercice 4 – Shell en C

A l'aide de la fonction `system`, faites un programme C affichant le contenu de votre répertoire personnel.

Exercice 5 – Débogage

Compilez le programme `code-mystere.c` en utilisant l'option `-g` de `gcc`, puis déboguez-le à l'aide de `gdb` jusqu'à atteindre l'exécution normale du programme.

Rappel des commandes gdb :
`break fichier:ligne` – *ajout d'un point d'arrêt dans le code*
`run arg1 arg2 ...` – *exécution du programme*
`CTRL + c` – *envoi d'un signal d'interruption au programme*
`next` – *exécution de l'instruction suivante*
`continue` – *reprise de l'exécution du programme*
`print var` – *affichage du contenu d'une variable*
`backtrace` – *affichage de la pile d'appels des fonctions*
`up/down i` – *remontée/descente de i dans la pile d'appels*
`quit` – *arrêt du débogueur*

Exercice 6 – Optionnel : Shell en C (bis)

Écrivez le programme C répondant aux questions de l'exercice 2.

IN405 – Système d'exploitation
TD 2 – Système de fichiers (1/2)

S. Gougeaud

2017/2018

L'archive `code-td2.tar` *contient trois fichiers :*

- `se_fichier.h` – *fichier d'en-têtes contenant les prototypes de la 'bibliothèque' à écrire ;*
- `main.c` – *fichier principal de l'exercice 2, contenant différents scénarios utilisant la 'bibliothèque' ;*
- `main-opt.c` – *fichier principal de l'exercice optionnel 3, contenant différents scénarios utilisant les fonctions avancées de la 'bibliothèque'.*

Exercice 1 – Bibliothèque de fonctions d'E/S

L'objectif de ce premier exercice est de créer une bibliothèque d'entrée/sorties utilisant les appels systèmes liés au système de fichier. A cet effet, il vous est demandé d'écrire le contenu du fichier `se_fichier.c`, composé du corps des fonctions énoncées dans `se_fichier.h`. **Attention : vous devez respecter les prototypes du fichier d'en-tête.**

Pour compiler le fichier `se_fichier.c` et en obtenir une bibliothèque, vous devez utiliser les commandes suivantes :

```
$ gcc -c -fPIC se_fichier.c
$ ar rcs libsef.a se_fichier.o
```

Les fonctions de la bibliothèque sont alors contenues dans l'archive `libsef.a`. Pour utiliser les fonctions d'une bibliothèque dans un autre programme, le compilateur doit avoir accès à deux ressources : les fonctions compilées (ici `libsef.a`) et les prototypes de fonctions apportées par le fichier d'en-tête (ici `se_fichier.h`). En supposant que ces ressources sont situées dans le même répertoire que votre programme issu de `test.c`, sa compilation se fait à l'aide de la commande suivante :

```
$ gcc test.c -L. -lsef
```

L'option `-L` permet d'indiquer au compilateur un chemin supplémentaire pour la recherche de bibliothèque ; le chemin indiqué est le répertoire courant. L'option `-l` lie la bibliothèque sef (`libsef.a`) au programme.

1. Ecrivez la fonction `SE_ouverture()` qui ouvre le fichier dont le chemin est donné en paramètre. Attention à bien créer le fichier s'il n'existe pas.
2. Ecrivez la fonction `SE_fermeture()` qui ferme le fichier donné en paramètre.
3. Ecrivez la fonction `SE_suppression()` qui supprime le fichier donné en paramètre.
4. Ecrivez la fonction `SE_lectureCaractere()` qui lit le prochain caractère du fichier donné en paramètre. Vérifiez si le fichier a été ouvert avec les bonnes permissions.
5. Ecrivez la fonction `SE_ecritureCaractere()` qui écrit le caractère dans le fichier donné en paramètre. Vérifiez si le fichier a été ouvert avec les bonnes permissions.

Exercice 2 – Utilisation de la bibliothèque

A l'aide des fonctions de votre bibliothèque, remplissez les corps des fonctions suivantes du fichier `main.c`. Le programme issu de la compilation consiste en l'exécution de tests vérifiant le bon fonctionnement de ces fonctions.

1. Ecrivez la fonction `affichage()` ayant le même comportement que la commande `cat`.
2. Ecrivez les fonctions `copie()` et `deplacement()` ayant respectivement le même comportement que les commandes `cp` et `mv`.
3. Ecrivez la fonction `sontIdentiques()` qui retourne `1` si les fichiers sont identiques, et `0` sinon.

Exercice 3 – Optionnel : Amélioration de la bibliothèque

Vous allez maintenant ajouter des fonctions de lecture/écriture à votre bibliothèque. Pour ceci, décommentez les prototypes restants de `se_fichier.h`, et écrivez les corps correspondants dans `se_fichier.c`. Ces fonctions peuvent être testées grâce au programme issu de `main-opt.c`.

1. Ecrivez les fonctions de lecture/écriture d'une chaîne de caractère.
2. Ecrivez les fonctions de lecture/écriture d'un entier (comportement de scanf avec %d).
3. Ecrivez la fonction de lecture d'un mot (comportement de scanf avec %s)

IN405 – Système d'exploitation
TD 3 – Système de fichiers (2/2)

S. Gougeaud

2017/2018

Exercice 1 – Méta-données des fichiers

En utilisant les fonctions `stat` et/ou `lstat`, écrivez les fonctions répondant aux comportements suivants :

1. Affichage du type de fichier : socket (sock), lien symbolique (link), fichier régulier (file), device (devc), répertoire (repy), FIFO (fifo).
2. Affichage des permissions d'accès au fichier : rwxrwxrwx.
3. Affichage du propriétaire du fichier.
4. Affichage de sa taille

Exercice 2 – Lecture d'un répertoire

En utilisant les fonctions `opendir` et `readdir`, affichez le contenu d'un répertoire donné.

Exercice 3 – Implémentation de la commande `ls`

A partir des codes des deux exercices précédents, écrivez une fonction affichant le contenu d'un répertoire, et indiquant pour chaque item, son type, ses permissions en écriture, son propriétaire et sa taille dans le format suivant :

```
type rwxrwxrwx owner size name1
type rwxrwxrwx owner size name2
```

Exercice 4 – Optionnel : Edition de fichier

En utilisant les appels système `mmap`, `msync` et `munmap`, écrivez un programme prenant en argument un chemin de fichier, afin de l'éditer. L'édition consistera en un remplacement de chaque voyelle du fichier par un caractère '*'.