TP 3

SYSTEMS D'EXPLOITATION

DÉLIVRABLES

- Un rapport PDF
- Code source commenté

- Phrases entières
- Code source commenté et fonctionnel
- Dossier à rendre : <Prenom1>.<Nom1>.<Prenom2>.<Nom2>TP3.zip(ou tar.gz)

INFORMATIONS GÉNÉRALES

- Références au cours :
  - 4. Language C
  - 5. Appels System
  - 6. Fichiers et répertoires
- Objectifs :
  - manipuler des fichiers;
  - obtenir et modifier les attributs d'un inode;
  - manipuler des liens;
  - parcourir des architectures de dossiers

EXERCICES

- Dans ce TP nous nous limiterons à la création d'un programme qui permettra de:
  - lister les fichiers d'un dossier de manière récursive;
  - copier le contenu d'un dossier dans un autre de manière récursive ;
  - le programme que nous développerons s'appellera ultra-cp (Makefile)

EXERCICES

Voici un exemple de sortie du programme:

```
> ultra-cp ../../TP4
drwxrwxr-x      4096 Mon Nov  9 10:09:47 2015  ../../TP4
drwxrwxr-x      4096 Thu Nov 19 14:56:59 2015  ../../TP4/doc
-rw-rw-r--       928 Thu Nov 19 14:56:24 2015  ../../TP4/doc/TP4.aux
-rw-rw-r--       152 Thu Nov 19 14:56:24 2015  ../../TP4/doc/TP4.out
-rw-rw-r--        85 Thu Nov 19 14:55:08 2015  ../../TP4/doc/Makefile
-rw-rw-r--    143314 Thu Nov 19 14:56:24 2015  ../../TP4/doc/TP4.pdf
-rw-rw-r--      6150 Thu Nov 19 14:56:59 2015  ../../TP4/doc/TP4.tex
-rw-rw-r--      6151 Thu Nov 19 14:55:48 2015  ../../TP4/doc/TP4.tex~
-rw-rw-r--     25261 Thu Nov 19 14:56:24 2015  ../../TP4/doc/TP4.log
drwxrwxr-x      4096 Thu Nov 19 14:17:51 2015  ../../TP4/src
-rw-rw-r--      8480 Thu Nov 19 14:02:33 2015  ../../TP4/src/main.o
-rw-rw-r--       358 Thu Nov 19 13:30:48 2015  ../../TP4/src/Makefile
-rw-rw-r--      1180 Thu Nov 19 11:11:29 2015  ../../TP4/src/options.h
-rw-rw-r--     10250 Thu Nov 19 14:17:27 2015  ../../TP4/src/proc_entries.c
-rw-rw-r--      1141 Thu Nov 19 14:14:04 2015  ../../TP4/src/proc_entries.h
-rw-rw-r--       395 Thu Nov 19 13:28:18 2015  ../../TP4/src/err.c
-rw-rw-r--       365 Thu Nov 19 13:28:55 2015  ../../TP4/src/err.h
-rw-rw-r--      6520 Thu Nov 19 14:01:27 2015  ../../TP4/src/err.o
-rw-rw-r--      2859 Thu Nov 19 14:02:30 2015  ../../TP4/src/main.c
-rw-rw-r--      7920 Thu Nov 19 14:01:27 2015  ../../TP4/src/options.o
-rw-rw-r--     14520 Thu Nov 19 14:17:51 2015  ../../TP4/src/proc_entries.o
-rw-rw-r--      2711 Thu Nov 19 13:24:45 2015  ../../TP4/src/options.c
-rwxrwxr-x     28259 Thu Nov 19 14:17:51 2015  ../../TP4/src/ultra-cp
```

EXERCICES

```
1 ultra-cp file1 folder1 folder2/ destination
```

aura pour effet de créer l'architecture suivante dans destination:

```
1 drwxrwxr-x      4096 Thu Nov 19 15:09:28 2015  destination
2 drwxrwxr-x      4096 Thu Nov 19 15:09:58 2015  destination/folder1
3 drwxrwxr-x      4096 Thu Nov 19 15:09:58 2015  destination/folder1/emptydir
4 -rw-rw-r--         0 Thu Nov 19 15:09:50 2015  destination/folder1/f2
5 -rw-rw-r--         0 Thu Nov 19 15:09:48 2015  destination/folder1/f1
6 drwxrwxr-x      4096 Thu Nov 19 15:10:06 2015  destination/folder2
7 -rw-rw-r--         0 Thu Nov 19 15:10:04 2015  destination/folder2/f5
8 -rw-rw-r--         0 Thu Nov 19 15:10:06 2015  destination/folder2/f4
9 -rw-rw-r--        14 Thu Nov 19 15:09:21 2015  destination/file1
```

Les fichiers et dossiers crées/modifiés auront les mêmes droits (si possible) que les sources. Question: dans quel cas les droits ne pourrons pas être les mêmes ?

EXERCICES

4.2 Modification de fichiers existants

Le programme pourra remplacer des fichiers éxistants mais ne devra jamais supprimer des fichiers/dossiers. Donc si un fichier est présent dans la destination mais pas dans la source il ne devra PAS être supprimé de la destination.

Si le fichier/dossier de destination éxiste déjà il devra être remplacé UNIQUEMENT si la taille de la source et de la destination sont différentes ou si la date de modification de la source est plus récente que celle de la destination.

EXERCICES

Lister les fichiers d'un dossier de manière récursive

La sortie est donc similaire à la fonction ls et elle contient:

- le type de fichier: d pour un dossier, - pour un fichier régulier, l pour un lien; d'autre peuvent être implémentés;
- les droits en lecture/écriture/exécution: les droits d'accès ne concernent que les droits "basiques" mentionnés; les bits setuid/setgid, sticky bit et autres ne seront pas représentés;
- la taille du fichier en octets;
- la date de modification: voir la fonction *strftime*;
- le nom du fichier: donné relativement au dossier/fichier passé en paramètre.

APPEL SYSTÈME `stat`

L'appel système `stat()` permet de garnir une structure stat:

```
int stat(const char *path, struct stat *buf);
```

La fonction retourne 0 si tout s'est bien passé ou -1 en cas d'erreur (cf. `errno`)

```
struct stat infos;
char *filename = "/tmp/foo.txt";
if( stat( filename, &infos ) < 0 )
    fprintf( stderr, "Cannot stat %s: %s\n", filename, strerror(errno) );
else
    printf( "Filesize: %d\n", infos.st_size );
```

DÉTERMINER LE TYPE D'UN INODE

- Le champ `st_mode` est un champs de bits contenant les permissions et le type d'un inode.
- Il existe plusieurs macro POSIX permettant de tester les types:

`S_ISREG(m)`	fichier de données ?
`S_ISDIR(m)`	répertoire ?
`S_ISCHR(m)`	character device ?
`S_ISBLK(m)`	block device ?
`S_ISFIFO(m)`	FIFO (named pipe) ?
`S_ISLNK(m)`	lien symbolique ?
`S_ISSOCK(m)`	socket?

```
if( S_ISDIR( info.st_mode ) ) {
    printf( "L'inode est un repertoire.\n" );
}
```

QUELQUES CONSEILS

```
//check file type
if(S_ISDIR(infos.st_mode))
      printf("d");
//check user permission
if(infos.st_mode & S_IRUSR)
      printf("r");
```

```
//print file size
 printf("%ld", infos.st_size);
```

- **lstat**() is identical to **stat**(), except that if *pathname* is a symbolic link, then it returns information about the link itself, not the file that it refers to.
- **fstat**() is identical to **stat**(), except that the file about which information is to be retrieved is specified by the file descriptor *fd*.

STRUCTURE `dirent`

Les entrées d'un répertoire sont représentées par la structure:

```
struct dirent {                 /* dirent.h */
    ino_t   d_ino;              /* inode number */
    off_t   d_off;              /* opaque value used to get next dirent (do not use) */
    unsigned short  d_reclen;   /* length of this record */
    unsinged char   d_type;     /* type of file; not supported by all file systems */
    char            d_name[256]; /* filename (NULL terminated), sometimes d_name[0] */
};
```

Pour accéder aux entrées d'un répertoire, il faut:

1. "Ouvrir" le répertoire avec `opendir()`
2. "Lire" l'entrée suivante avec `readdir()`
3. Répéter 2, jusqu'à épuisement des entrée ou tout autre critère
4. "Fermer" le répertoire avec `closedir()`

QUELQUES CONSEILS

```
#include <dirent.h>

void RecursiveFunc(char *entry)
{
        struct dirent *dirbase;
        DIR *dir = opendir(entry);
        while ((dirbase = readdir(dir)) != NULL){
                //do your stuff
                 strcmp();
                // Construct subfolder path and update
                strcpy();
                strcat();

                RecursiveFunc(newpath);
        }
        closedir();

                                dirbase->d_name
}
                  Attention: be careful with “.” and “..”
```

MAKEFILE

mymain.c	myfunc.c	mymain.h

#include “mymain.h”
int main() {
// call a function in another file
myGreatFunc(); return(0); }
	#include <stdio.h>
	#include “mymain.h”
	void myGreatFunc (void)
	{
	printf("Hello world!\n");
	return;
	}
		/* example include file */ void myGreatFunc (void);

```
gcc -o mymain mymain.c myfunc.c
```

Makefile

version1

```
mymain: mymain.c myfunc.c
TAB gcc -o mymain mymain.c myfunc.c
```

Makefile

version2

```
CC=gcc
CFLAGS=-g
mymain: mymain.o myfunc.o
TAB $(CC) -o hellomake hellomake.o hellofunc.o:
```

MAKEFILE

```
CC = gcc
CFLAGS = -g -c -I
LFLAGS = -lssl -lcrypto
OBJECTS = mymain.o myfunc.o
EXECUTABLE = ultra-cp

all: $(EXECUTABLE)

$(EXECUTABLE): $(OBJECTS)
        $(CC) $(OBJECTS) -o $(EXECUTABLE)
$(LFLAGS)

%.o: %.c
        $(CC) $(CFLAGS) $< -o $@

clean:
        rm $(OBJECTS) $(EXECUTABLE)
```

version3

gcc --help