Rapport TP Système d’exploitation : SE

TP2

Binôme : Ferchichi Manel

Madi Sara

Section : ISIL B

Groupe : 2 et 3

L'objectif de ce TP est la manipulation de fichiers et répertoires à l'aide des appels systèmes et des fonctions de la bibliothèque C.
Placez-vous dans le répertoire TP2.

: Création de répertoire TP2 :

- **mkdir TP2**
- **cd TP2**

```
ubuntu@ubuntu:~$ mkdir TP2
ubuntu@ubuntu:~$ cd TP2
ubuntu@ubuntu:~/TP2$
```

Exercice 1:

1. Avec un éditeur de texte (Ex. gedit), créez un fichier creat.c contenant le programme C:
Après la création du répertoire TP2 on crée le file creat.c dans ce répertoire avec :

méthode une :

- touch creat.c
- nohup gedit & creat.c

méthode deux :

- nohup gedit & creat.c

```
ubuntu@ubuntu:~/TP2$ nohup gedit creat.c
nohup: ignoring input and appending output to 'nohup.out'
```

2. Interprétez les différentes instructions de ce code. Déduisez le rôle de ce programme.
on remplit le file avec le programme donné :
Interprétation :

//les bibliothèques utilisées :

#include <stdio.h> :l'inclusion des fonctions d'entrées/sorties à la bib standard
#include <sys/types.h> : elle définit les types de données used dans la source system
#include <sys/stat.h> :elle définit la structure de données
#include <fcntl.h>: pour l'utilisation de open()/elle définit aussi les arguments d entrées des fonctions included in this bib

```
#include <stdlib.h> :  propose des fonctions pour répondre à
différentes problématiques



//code

/*déclaration de la fonction main avec deux variables d
entrées
argc un entier
argv est un tableau de type char*\
int main(int argc, char *argv[]) {
int fd; //déclaration d'un entier

//cette partie vérifie le nombre d'arguments de la fonction si
il dépasse un seul argument elle déclenche une erreur puis
exit
if (argc != 2) {
printf("Usage: %s <File name#>\n", argv[0]);
exit(1);
}

//on initialise les droit d accès avec des macros comme suit
:  S_IRUSR | S_IWUSR | S_IXUSR ⇒ pour l initialisation des
droits (read , write , execute for user )
: S_IRGRP | S_IWGRP ⇒ pour l initialisation des  droits (read
, write ) for group
:S_IROTH | S_IWOTH ⇒pour l initialisation des  droits (read ,
write ) for others
mode_t mode = S_IRUSR | S_IWUSR | S_IXUSR | S_IRGRP | S_IWGRP
| S_IROTH | S_IWOTH;

//création et ouvrir le fichier avec les droit précédente
//fd = 0 means no error for open
//fd = -1 error of open
fd = open(argv[1], O_WRONLY | O_EXCL | O_CREAT, mode);

//si la fonction ne s'exécute pas proprement déclencher
perror et sortir
if (fd == -1) {
perror("open");
exit(-1);
}

//sinon sortir avec succès
return EXIT_SUCCESS;
}
```

---

3. Créez l'exécutable creat et exécutez-le en donnant comme argument en entrée fich1.txt.
création du file plus remplir le file :

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdlib.h>
int main (int argc , char *argv[]) {
    int fd ;
    if (argc != 2) {
                    printf("Usage : %s <File name #>\n", argv[0]);
                    exit(1);
    }

    mode_t mode =S_IRUSR | S_IWUSR | S_IXUSR | S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH;

    fd = open (argv[1] , O_WRONLY | O_EXCL | O_CREAT , mode );

    if (fd ==-1){
        perror("open");
        exit(-1);
    }

    return EXIT_SUCCESS;
}
```

execution of file avec :
compilation

- gcc creat.c -o creat

exécution

- ./creat fich1.txt

```
ubuntu@ubuntu:~/TP2$ gcc creat.c -o creat
ubuntu@ubuntu:~/TP2$ ./creat fich1.txt
```

4. Refaites l'exécution de ce programme en donnant encore une fois fich1.txt comme argument. Que remarquez-vous ?

compilation

- `gcc creat.c -o creat`

exécution

- `./creat fich1.txt`

```
ubuntu@ubuntu:~/TP2$ gcc creat.c -o creat
ubuntu@ubuntu:~/TP2$ ./creat fich1.txt
open: File exists
```

on remarque que :
fich1.txt existe déjà⇒a la première exécution ce fichier a été créé à l'aide la fonction open avec les droits accès données
lors de la deuxième exécution la fonction open ⇒retourne -1 la cause : existence de fichier impossible de créer un fichier with same name

5. Quelle est la valeur du masque par défaut des droits d'accès ?

- `umask`

```
ubuntu@ubuntu:~/TP2$ umask
0002
```

6. Listez les droits d'accès du fichier fich1.txt.
les droits d' accès sont :
762

```
ubuntu@ubuntu:~/TP2$ ls -l  fich1.txt
-rwxrw-r-- 1 ubuntu ubuntu 0 Dec  8 02:55 fich1.txt
ubuntu@ubuntu:~/TP2$
```

7. Comparez ces droits à ceux de la variable mode du programme creat.c. Quelle relation existe-t-elle entre eux ?

- les droits d'accès de file creat.c

```
ubuntu@ubuntu:~/TP2$ ls -l creat.c
-rw-rw-r-- 1 ubuntu ubuntu 531 Dec  8 02:54 creat.c
ubuntu@ubuntu:~/TP2$
```

- il y a pas forcément une relation précise à part une relation d'inclusion

exercice 2:

1. Quel est le rôle de l'appel système stat() ?
la commande stat() est un appel système qui retourne les information d' un fichier saisi (fichier system)

- les informations qu' elle donne :
- taille : droit (read write execute) :id (groupe or user) :num inode etc

2. Avec un éditeur de texte (Ex. gedit), créez un fichier attributsInode.c contenant le programme C suivant :

création d'un fichier

méthode une :
- **touch attributsInode.c**
- **nohup gedit & creat.c**

méthode deux :
- **nohup gedit & attributsInode.c**

3. Interprétez les différentes instructions de ce code. Déduisez le rôle de ce programme.

code

---

interprétation du code

```
//pour les bibliothèque la même chose que l' exercice 1
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h> //sauf
#include <pwd.h>//elle contient plusieur information of struct  :id , user , password etc

int main(int argc, char *argv[]) {//déclaration de la fonction avec les arguments
struct stat StatusBuffer; //création d' un enregistrement qui contiendra par suite toute les information stat of the argument
```

```
//remplir les informations d' argument dans l' enregistrement si il
n'existe pas alors erreur et exit
if (stat(argv[1], &StatusBuffer) == -1) {
perror(argv[1]);
exit(1);
}



//affichage du nom de fichier entré
printf("Nom de fichier : %s\n",argv[1]);

//affichage des droits accès en mode octal du fichier qui est dans
le champ (StatusBuffer.st_mode &0777)
printf("Droits d'accès (mode octal) : %o\n",StatusBuffer.st_mode
&0777);

vérification des la valeur d'attribut st_mode grâce au macron
S_ISREG⇒fichier régulier
S_ISDIR⇒repertoire
donc affichage du type
-----------------------------------------------
if (S_ISREG(StatusBuffer.st_mode)) {
printf("%s est un fichier régulier\n", argv[1]);
} else if (S_ISDIR(StatusBuffer.st_mode)) {
printf("%s est un répertoire\n", argv[1]);
};
-----------------------------------------------

vérification des accès de fichier ou répertoire et comparer avec
celle de st_mode si c est équivalent on les affiche sinon afficher
rien
-----------------------------------------------
printf ("Les droits d'accès de USER en mode symbolique sont :");
printf ("%c", StatusBuffer.st_mode & S_IRUSR ? 'r' : '-');
printf ("%c", StatusBuffer.st_mode & S_IWUSR ? 'w' : '-');
printf ("%c\n", StatusBuffer.st_mode & S_IXUSR ? 'x' : '-');
-----------------------------------------------
//afficher ID of the user (name)
printf("Owner: %s\n",getpwuid(StatusBuffer.st_uid)->pw_name);

//afficher la taille du fichier ou répertoire
printf("Taille: %ld\n",StatusBuffer.st_size);
return EXIT_SUCCESS;
}
```

---

4. Créez l'exécutable attributsInode et exécutez-le.
creation faite .
remplir le fichier en C puis l'exécuter

le code :

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <pwd.h>

int main (int argc , char *argv[]){

struct stat StatusBuffer ;

if (stat(argv[1] ,&StatusBuffer)==-1){
perror(argv[1]);
exit(1);
}

printf("Nom de fichicer : %s \n " , argv[1]);
printf("droits d accès  (mode octal) : %o \n",StatusBuffer.st_mode
&0777);

if (S_ISREG (StatusBuffer.st_mode)){
printf("%s est un fichier regulier \n ",argv[1]);


}else if (S_ISDIR (StatusBuffer.st_mode)){
printf("%s est un repertoir \n ",argv[1]);
}
```

```c
printf("les droits d'accès de USER en mode symbolique sont : ") ;
printf("%c",StatusBuffer.st_mode & S_IRUSR ? 'r' : '-');
printf("%c",StatusBuffer.st_mode & S_IWUSR ? 'w' : '-');
printf("%c\n",StatusBuffer.st_mode & S_IXUSR ? 'x' : '-');

printf("owner : %s \n",getpwuid(StatusBuffer.st_uid)->pw_name);
printf("taille : %ld \n",StatusBuffer.st_size);

return EXIT_SUCCESS ;
}
```

l exécution :

```
ubuntu@ubuntu:~/TP2$ gcc attributsInode.c -o attributsIno
de
ubuntu@ubuntu:~/TP2$ ./attributsInode tpsys
```

résultat :
cas existant :
répertoire :

```
ubuntu@ubuntu:~/TP2$ ./attributsInode tpsys
Nom de fichicer : tpsys
 droits d accès  (mode octal) : 775
tpsys est un repertoir
 les droits d'accès de USER en mode symbolique sont : rwx
owner : ubuntu
taille : 100
```

fichier:

```
ubuntu@ubuntu:~/TP2$ ./attributsInode attributsInode.c
Nom de fichicer : attributsInode.c
 droits d accès  (mode octal) : 664
attributsInode.c est un fichier regulier
 les droits d'accès de USER en mode symbolique sont : rw-
owner : ubuntu
taille : 915
```

cas non existant :

```
ubuntu@ubuntu:~/TP2$ ./attributsInode attribut
attribut: No such file or directory
```

5. Modifiez ce programme afin de simuler l'exécution de la commande ls -l sur le fichier passé en paramètres.

droit accès the user then date and hour then file name
le code :

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <pwd.h>
#include <time.h>

char* formatdate(char* str , time_t val ){
strftime(str,36,"%d.%m.%Y.%H.%M:%s",localtime(&val));
}

int main (int argc , char *argv[]){

struct stat StatusBuffer ;

if (stat(argv[1] ,&StatusBuffer)==-1){
perror(argv[1]);
exit(1);
}

char date[36];

/*
if (S_ISREG (StatusBuffer.st_mode)){
printf("%s est un fichier regulier \n ",argv[1]);

}else if (S_ISDIR (StatusBuffer.st_mode)){
printf("%s est un repertoir \n ",argv[1]);
}
*/
//user
printf("%c",StatusBuffer.st_mode & S_IRUSR ? 'r' : '-');
printf("%c",StatusBuffer.st_mode & S_IWUSR ? 'w' : '-');
printf("%c",StatusBuffer.st_mode & S_IXUSR ? 'x' : '-');
//group
printf("%c",StatusBuffer.st_mode & S_IRGRP ? 'r' : '-');
printf("%c",StatusBuffer.st_mode & S_IWGRP ? 'w' : '-');
printf("%c",StatusBuffer.st_mode & S_IXGRP ? 'x' : '-');
//others
printf("%c",StatusBuffer.st_mode & S_IROTH ? 'r' : '-');
printf("%c",StatusBuffer.st_mode & S_IWOTH ? 'w' : '-');
printf("%c",StatusBuffer.st_mode & S_IXOTH ? 'x' : '-');
//date appel fonction
time_t t ;
time(&t);
printf("%s",ctime(&t));
//owner taille nom fichier
printf(" %s ",getpwuid(StatusBuffer.st_uid)->pw_name);
printf(" %ld ",StatusBuffer.st_size);
printf("%s   " , argv[1]);
printf(" %o \n",StatusBuffer.st_mode &0777);
```

exécution :

```
ubuntu@ubuntu:~/TP2$ gcc attributsInode.c -o attributsInode
ubuntu@ubuntu:~/TP2$ ./attributsInode attributsInode.c
rw-rw-r--Thu Dec  8 05:49:53 2022
 ubuntu  1378 attributsInode.c    664
```

exercice 3:

Écrire un programme C qui copie le contenu d'un fichier source vers un fichier destination, en simulant la commande cp.

open file with gedit then écrire mon code :

```c
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>
int main (int argc , char *argv[]) {
//declarations des variables
    char buf;
    int fdsource;
    int fddest;

//verificatins de nombre d'arguments  ; si c est different  a 3 alors error si c est egale a 1 alors c est un manque d arguments si argument 1 et 2 sont egaux alors affiche ce message
    if (argc != 3) {
                    printf("Usage : %s <File name #>\n", argv[0]);
                    exit(1);
    }
    if (argc == 1) {
                    printf("manque darguments pour la fonction \n");
                    exit(1);
    }

    if (argv[1]==argv[2]){
                   printf("les fichier sont egaux (les memes )\n");
                   exit(1);
```

```
//si tt les conditions sont verifiées alors ouvrir file source et
destination , source read only et destination write only

    fdsource = open (argv[1] ,O_RDONLY );
    fddest = open (argv[2] ,O_WRONLY);

//verification si open s est executé  correctement
    if (fdsource ==-1){
        printf("fichier source n existe pas\n ");
        exit(-1);
    }
    if (fddest ==-1){
        printf("fichier source n existe pas donc on le creer \n");
        //si file source n existe pas alors creation de ce dernier
mode_t mode =S_IRUSR | S_IWUSR | S_IXUSR | S_IRGRP |  S_IWGRP |
S_IROTH | S_IWOTH;


       fddest = open (argv[2] , O_WRONLY | O_EXCL | O_CREAT ,
mode );

    }
    //remplissage de ficher destination avec source
    while( read(fdsource,&buf,1)!=0 ){
```

```
    }
    //remplissage de ficher destination avec source
    while( read(fdsource,&buf,1)!=0 ){
    write(fddest,&buf,1);
    printf("\n");

    }
    //closing both files
    close(fdsource);
    close(fddest);



    printf("done with success");
    return EXIT_SUCCESS;
}
```

exécution du file :

```
ubuntu@ubuntu:~/TP2$ gedit exercice3.c
ubuntu@ubuntu:~/TP2$ gcc exercice3.c -o exercice3
ubuntu@ubuntu:~/TP2$ ./exercice3 creat.c test3.txt
fichier source n existe pas donc on le creer
```

result:

```
done with successubuntu@ubuntu:~/TP2$
```

d’autre exécution :

```
ubuntu@ubuntu:~/TP2$ gcc exercice3.c -o exercice3
ubuntu@ubuntu:~/TP2$ ./exercice3 creat.c
Usage : ./exercice3 <File name #>
ubuntu@ubuntu:~/TP2$ gcc exercice3.c -o exercice3
ubuntu@ubuntu:~/TP2$ ./exercice3 ct.c test3.txt
fichier source n existe pas
 ubuntu@ubuntu:~/TP2$gcc exercice3.c -o exercice3
ubuntu@ubuntu:~/TP2$ ./exercice3
Usage : ./exercice3 <File name #>
```

exercice 4:

question 1:

code avec des commentaire dans les capture :(interprétation)

```c
#include <stdio.h>
#include <dirent.h>
#include <stdlib.h>
#include <string.h>

int main(int argc , char *argv[]){

DIR *rep; //rep pointeur sur le type dir
struct dirent *dir; //dir structure de typedirent , dirent c est le type de dir

//verifier c est le nombre d arguments est un seul nom de repertoire
if (argc>2){
printf("veuillez entrer un seul nom repertoire svp!");
exit(1);

//si le nombre dargument est un seul
//cas aucun repertoire specifié donc on affiche le contenu du repertoire courant
}else if (argc==1){

//le point de  chaine signifie : repertoire courant
rep=opendir(".");
```

```c
//message affichage du nom repertoire puis son contenu
printf("le contenu du repertoire courant est:");
while ((dir=readdir(rep))!=NULL){

//on fait une condition pour eviter l affichage des points
if((strcmp(dir-> d_name,".")!=0) && (strcmp(dir-> d_name,"..")!=0)){
//d_name est un champ de structure dir qui contier le nom de fichier
inside repertoire

printf("%s\n",dir->d_name);

}

}
closedir(rep);
}


else if (argc==2){
rep=opendir(argv[1]);
```

```c
//cas repertoire n existe pas
if (rep==NULL){
//FAIL TO OPEN REPERTOIRE OU IL N EXISTE PAS
printf("le repertoire n existe pas !!");
}
//cas de repertoire entré est existant
else {
//message pour le nom de repertoire a afficher son contenu
printf ("le contenu du repertoire %s est :", argv[1]);
while ((dir=readdir(rep))!=NULL)
printf("%s\n",dir->d_name);
closedir(rep);
}
}

return EXIT_SUCCESS;

}
```

compilation et exécution du code:

```
ubuntu@ubuntu:~/TP2$ gcc exercice4.c -o exercice4
ubuntu@ubuntu:~/TP2$ ./exercice4
```

cas non répertoire donné

```
ubuntu@ubuntu:~/TP2$ ./exercice4
le contenu du repertoire courant est:exercice4
exercice4.c
exercice3
exercice3.c
test3.txt
creat.c
fich2.txt
creat
fich1.txt
nohup.out
```

cas répertoire inexistant :

```
ubuntu@ubuntu:~/TP2$ ./exercice4 hello
le repertoire n existe pas !!ubuntu@ubuntu:~/TP2$
```

cas arguments>2 :

```
ubuntu@ubuntu:~/TP2$ gcc exercice4.c -o exercice4
ubuntu@ubuntu:~/TP2$ ./exercice4 kkkk kkk llll
veuillez entrer un seul nom repertoire svp!ubuntu@ubuntu:
~/TP2$
```

cas success :

```
ubuntu@ubuntu:~/TP2$ mkdir tpsys
ubuntu@ubuntu:~/TP2$ cd tpsys
ubuntu@ubuntu:~/TP2/tpsys$ touch manel.txt
ubuntu@ubuntu:~/TP2/tpsys$ touch hello.c
ubuntu@ubuntu:~/TP2/tpsys$ mkdir bye
ubuntu@ubuntu:~/TP2/tpsys$ cd ..
ubuntu@ubuntu:~/TP2$ gcc exercice4.c -o exercice4
ubuntu@ubuntu:~/TP2$ ./exercice4 tpsys
le contenu du repertoire tpsys est :.
..
bye
hello.c
manel.txt
```

question 2: