

# Systeme 1

## TD n°2 : Entrées/sorties

### Exercice 1 Liens symboliques

On considère le programme liens.c suivant :

```
#include <sys/stat.h>
int main(int argc, char **argv){
    struct stat s1, s2, s3, s4 ;
    argv++ ;
    if (link(*argv,"lien")==-1) printf("La fonction link a mal fonctionne\n") ;
    else printf("La fonction link a bien fonctionne\n") ;
    if (symlink(*argv,"liensymb")==-1) printf("La fonction symlink a mal fonctionne\n") ;
    else printf("La fonction symlink a bien fonctionne\n") ;
    system("ls -lai");
    /* appel a la fonction stat et lstat */
    if (stat(*argv,&s1)==-1) printf("La fonction stat a mal fonctionne\n") ;
    else printf("L'inode du fichier %s est %lu\n",*argv,s1.st_ino) ;
    if (stat("lien",&s2)==-1) printf("La fonction stat a mal fonctionne\n") ;
    else printf("L'inode du lien est %lu\n",s2.st_ino) ;
    if (stat("liensymb",&s3)==-1) printf("La fonction stat a mal fonctionne\n") ;
    else printf("L'inode du lien liensymb est %lu\n",s3.st_ino) ;
    if (lstat("liensymb",&s4)==-1) printf("La fonction lstat a mal fonctionne\n") ;
    else printf("L'inode du lien liensymb est %lu\n",s4.st_ino) ;
    return 0 ;
}
```

On suppose que le répertoire contenant le programme liens.c contient seulement un deuxième fichier appelé fich.

1. Le fichier liens.c est compilé (gcc -o liens liens.c), puis exécuté à l'aide de la commande ./liens fich. Donner une trace possible de cette exécution.

-> fichier liens.c

```
> ls
fich liens liens.c
> ./liens fich
La fonction link a bien fonctionne
La fonction symlink a bien fonctionne
total 12
42106936 -rw-r--r-- 2 borne ocad 0 2008-03-18 13:05 fich
42106936 -rw-r--r-- 2 borne ocad 0 2008-03-18 13:05 lien
42106940 -rwxr-xr-x 1 borne ocad 8114 2008-03-18 13:23 liens
```

```

42106938 -rw-r--r-- 1 borne ocad 1851 2008-03-18 13:23 liens.c
42106937 lrwxrwxrwx 1 borne ocad 4 2008-03-18 13:25 liensymb -> fich
L'inode du fichier fich est 42106936
L'inode du lien est 42106936
L'inode du lien liensymb est 42106936
L'inode du lien liensymb est 42106937

```

Faire remarquer

- les similitudes entre fich et lien (même taille, même inode, mêmes droits, ...).
- les droits en exécution de l'exécutable créé lors de la compilation
- les particularités du lien symbolique (inode diff) (+ taille 4 alors que fichier pointé de taille nulle).

2. Que se passe-t-il si on exécute une nouvelle fois cette commande ? Pourquoi ?

```

> ./liens fich
La fonction link a mal fonctionne
La fonction symlink a mal fonctionne
total 12
42106936 -rw-r--r-- 2 borne ocad 0 2008-03-18 13:05 fich
42106936 -rw-r--r-- 2 borne ocad 0 2008-03-18 13:05 lien
42106940 -rwxr-xr-x 1 borne ocad 8114 2008-03-18 13:23 liens
42106938 -rw-r--r-- 1 borne ocad 1851 2008-03-18 13:23 liens.c
42106937 lrwxrwxrwx 1 borne ocad 4 2008-03-18 13:25 liensymb -> fich
L'inode du fichier fich est 42106936
L'inode du lien est 42106936
L'inode du lien liensymb est 42106936
L'inode du lien liensymb est 42106937

```

link et symlink n'ont pas fonctionné car les liens existaient déjà.

3. Compléter le programme en renommant lien en lien\_nouv et liensymb en liensymb\_nouv.

```

/* renommage des liens */
if (rename("lien","lien_nouv")==-1)
    printf("La fonction rename a mal fonctionne\n") ;
else printf("La fonction rename a bien fonctionne\n") ;

if (rename("liensymb","liensymb_nouv")==-1)
    printf("La fonction rename a mal fonctionne\n") ;
else printf("La fonction rename a bien fonctionne\n") ;

```

Que donne alors le `ls -li` ?

```

> ls
fich lien liens liens.c liensymb
> ./liens fich
La fonction link a mal fonctionne
La fonction symlink a mal fonctionne
total 16
42106936 -rw-r--r-- 2 borne ocad 0 2008-03-18 13:05 fich

```

```

42106936 -rw-r--r-- 2 borne ocad    0 2008-03-18 13:05 lien
42106940 -rwxr-xr-x 1 borne ocad 8428 2008-03-18 13:28 liens
42106938 -rw-r--r-- 1 borne ocad 1845 2008-03-18 13:28 liens.c
42106937 lrwxrwxrwx 1 borne ocad    4 2008-03-18 13:25 liensymb -> fich
L'inode du fichier fich est 42106936
L'inode du lien est 42106936
L'inode du lien liensymb est 42106936
L'inode du lien liensymb est 42106937
La fonction rename a bien fonctionne
La fonction rename a bien fonctionne
total 16
42106936 -rw-r--r-- 2 borne ocad    0 2008-03-18 13:05 fich
42106936 -rw-r--r-- 2 borne ocad    0 2008-03-18 13:05 lien_nouv
42106940 -rwxr-xr-x 1 borne ocad 8428 2008-03-18 13:28 liens
42106938 -rw-r--r-- 1 borne ocad 1845 2008-03-18 13:28 liens.c
42106937 lrwxrwxrwx 1 borne ocad    4 2008-03-18 13:25 liensymb_nouv -> fich

```

Pas de changement de inode, seul le nom change.

4. Terminer maintenant le programme en supprimant les liens qui ont été créés.

```

/* suppression des liens */
if (unlink("lien_nouv")==-1)
    printf("La fonction unlink a mal fonctionne\n") ;
else printf("La fonction unlink a bien fonctionne\n") ;

if (unlink("liensymb_nouv")==-1)
    printf("La fonction unlink a mal fonctionne\n") ;
else printf("La fonction unlink a bien fonctionne\n") ;

```

## Exercice 2 Entrées/sorties sur répertoires

Exercice pris sur le site [http://www.labri.fr/perso/gimbert/enseignement/iup1/iup1\\_03\\_04.html](http://www.labri.fr/perso/gimbert/enseignement/iup1/iup1_03_04.html)

```

>ls -R A
A:
B C fich

```

```

A/B:
fich2

```

```

A/C:

```

1. Rédiger une fonction `int ls(const char *ref)`; qui affiche à l'écran le contenu du répertoire `ref`. Cette fonction, comme les suivantes, devra retourner 0 en cas de succès et -1 en cas de problème.

-> fichier `ls.c`

```

>gcc -o exe ls.c

```

```
> ./exe A
B
C
fich
..
.
> ./exe fich
echec
```

2. À quoi sert la fonction `concat` suivante :

```
char *concat(const char *ref1,const char *ref2)
{
    int i,taille1=0,taille2=0;
    char *resultat;

    while(ref1[taille1]!='\0') taille1++;
    while(ref2[taille2]!='\0') taille2++;
    resultat=(char *)malloc((taille1+taille2+2)*sizeof(char));
    for (i=0;i<taille1;i++) resultat[i]=ref1[i];
    resultat[taille1]='/';
    for (i=0;i<=taille2;i++) resultat[taille1+i+1]=ref2[i];
    return resultat;
}
```

retourne la chaîne de caractères `ref1/ref2`

3. Rédiger une fonction `int ls_taille(const char *ref)`; qui affiche à l'écran la liste des entrées du répertoire `ref`, ainsi que leur taille. (Utiliser la fonction `concat`.)  
Ecrire une deuxième version `ls_taille_reg` qui n'affiche que les entrées qui sont des fichiers réguliers. (Rappel : Un fichier est régulier si la valeur `val` du champ `st_mode` de sa structure `stat` est telle que `val & S_IFMT==S_IFREG`.)

-> fichier `ls_taille.c`

```
> gcc -o exe ls_taille.c
> ./exe A
ls_taille
4096 B
4096 C
0 fich
4096 ..
4096 .
ls_reg
0 fich
```

4. Utiliser la récursivité pour afficher les fichiers réguliers de tous les sous répertoires.

-> fichier `ls_rec.c`

```
> gcc -o exe ls_rec.c
```

```
> ./exe A
0 A/B fich2
0 A fich
```

### Exercice 3 Projection des fichiers en mémoire

Un fichier projeté en mémoire apparaît comme un tableau d'octets, ce qui permet de le parcourir en tous sens plus commodément qu'avec des `seek`, `read` et `write`.

La fonction `mmap` permet d'associer une adresse en mémoire à un segment de fichier d'une certaine longueur. Un appel de la fonction

```
#include <unistd.h>
#include <sys/mman.h>
void * mmap(void *adr, int lg, int prot, int options, int descr, int offset);
```

permet de demander la projection à l'adresse `adr` du segment de fichier identifié par le descripteur `descr` commençant à la position `offset` par rapport au début du fichier et de longueur `lg`. L'adresse `adr` doit faire partie des adresses non utilisées par le processus : il est conseillé de transmettre l'adresse `NULL` qui entraîne le choix de l'adresse par le système (la valeur retournée est dans tous les cas l'adresse de projection du fichier). Le paramètre `prot` définit les protections appliquées à la zone mémoire et est construit comme combinaison bit à bit des constantes suivantes :

Constante	Interprétation
<code>PROT_NONE</code>	aucun accès autorisé
<code>PROT_READ</code>	accès en lecture autorisé
<code>PROT_WRITE</code>	accès en écriture autorisé
<code>PROT_EXEC</code>	exécution autorisée

Le paramètre `options` a les valeurs possibles suivantes :

Constante	Interprétation
<code>MAP_SHARED</code>	les modifications de l'objet sont partagées
<code>MAP_PRIVATE</code>	les modifications de l'objet sont faites sur une copie privée

Un appel à la primitive

```
#include <unistd.h>
#include <sys/mman.h>
int munmap(void *adr, int lg);
```

correspond à une demande de libération de la zone mémoire d'adresse `adr` et de longueur `lg`.

En utilisant la projection des fichiers en mémoire, écrire un programme C qui permet d'afficher les lignes d'un fichier en partant de la fin.

-> fichier `affich_envers.c`