STI 1ère année – Programmation Système

TD 5: Verrou - Projection Mémoire

J. Briffaut - J.-F. Lalande - C. Toinard

1 Verrou

1.1 Verrouillage et deadlock

Exercice 1 Créez les macros `read_lock`, `readw_lock`, `write_lock` et `writew_lock` permettant de poser des verrous de façon non bloquante ou bloquante en lecture et en écriture. Ces macros utilisent la même fonction C que vous écrivez et dont le prototype est :

```
int lock_reg (int fd, int lockCmd, int lockType, off_t regRffset, int
 regWhence, off_t regLen);
```

Vous devrez utiliser et construire une structure `flock` et la fonction `fcntl` dans la fonction `lock_reg`.

Listing 1 – Solution de l'exercice 1

```
#ifndef __lock_h
#define __lock_h
#include <sys/types.h>
#include <unistd.h>
#include <fcntl.h>

int
lock_reg(int fd, int cmd, int type, off_t offset, int whence
     , off_t len)
{
  struct flock    lock;

  lock.l_type = type;             /* F_RDLCK, F_WRLCK,
       F_UNLCK */
  lock.l_start = offset;  /* byte offset, relative to
       l_whence */
  lock.l_whence = whence; /* SEEK_SET, SEEK_CUR, SEEK_END */
  lock.l_len = len;               /* #bytes (0 means to EOF)
       */
  return( fcntl(fd, cmd, &lock) );
}

#define read_lock(fd, offset, whence, len) \
                        lock_reg(fd, F_SETLK, F_RDLCK,
                             offset, whence, len)
#define readw_lock(fd, offset, whence, len) \
                        lock_reg(fd, F_SETLKW, F_RDLCK,
                             offset, whence, len)
#define write_lock(fd, offset, whence, len) \
                        lock_reg(fd, F_SETLK, F_WRLCK,
                             offset, whence, len)
#define writew_lock(fd, offset, whence, len) \
                        lock_reg(fd, F_SETLKW, F_WRLCK,
                             offset, whence, len)
#define un_lock(fd, offset, whence, len) \
                        lock_reg(fd, F_SETLK, F_UNLCK,
                             offset, whence, len)

#endif
```

Exercice 2 Ecrivez un programme `lockfile` qui crée un fichier appelé *templock1* vide accessible en lecture et écriture pour le propriétaire, lecture pour le groupe et les autres. Puis il écrit la chaîne "ab" dans le fichier.

Exercice 3 Ajoutez une fonction `static void lockabyte(const char *name, int fd, off_t offset)` qui pose un verrou bloquant en écriture sur le fichier identifié par un descripteur de fichier `fd` à l'offset donné. `name` correspond au nom du programme appelant est sera utilisé pour afficher des informations sur la prise du verrou.

Exercice 4 Ajoutez à votre programme `lockfile` un appel à `lockabyte` puis utilisez la fonction `sleep()` pour faire attendre votre programme 30 seconde avant de libérez le verrou.

Listing 2 – Solution de l'exercice 4

```
#include        <sys/stat.h>
#include        <stdlib.h>
#include        <stdio.h>
#include        "lock.h"

#define FILE_MODE       (S_IRUSR | S_IWUSR | S_IRGRP |
      S_IROTH)

static void lockabyte(const char *, int, off_t);

int
main(void)
{
        int             fd;
                /* Create a file and write two bytes to it
                   */
        if ( (fd = creat("templock1", FILE_MODE)) < 0)
                perror("creat error");
        if (write(fd, "ab", 2) != 2)
                perror("write error");

        lockabyte("child", fd, 0);
        sleep(30);

        exit(0);
}

static void
lockabyte(const char *name, int fd, off_t offset)
{
        int ret=writew_lock(fd, offset, SEEK_SET, 1);
        if ( ret < 0)
        {
                perror(" writew_lock error");
        }

        printf("%s: got the lock, byte %d\n", name, (int)
              offset);
}
```

Exercice 5 Ecrire un second programme `trytolockfile` qui ouvre le fichier *templock1* en lecture et qui essai de poser un verrou en lecture. Testez vos de programme en lançant dans une première console `lockfile`, puis lancer `trytolockfile` dans une seconde console.

Listing 3 – Solution de l'exercice 5

```
#include        <sys/stat.h>
#include        <stdlib.h>
#include        <stdio.h>
#include        "lock.h"

static void lockabyte(const char *, int, off_t);

int
main(void)
{
        int             fd;

        if ( (fd = open("templock1", O_RDONLY)) < 0)
                perror("open ro error");

        lockabyte("prog2", fd, 0);

        exit(0);
}

static void
lockabyte(const char *name, int fd, off_t offset)
{
        int ret=read_lock(fd, offset, SEEK_SET, 1);
        if ( ret < 0)
        {
                perror("read_lock error");
        }

        printf("%s: got the lock, byte %d\n", name, (int)
              offset);
}
```

Exercice 6 Modifier les programmes `lockfile` et `trytolockfile` afin de tester les différentes combinaisons de verrous bloquant/non bloquant en lecture/ecriture.

2 Projection d'un fichier

Exercice 7 Ecrivez un programme qui projette en mémoire un segment de mémoire partagé qui correspond au fichier "test.dat". Ce fichier contient des caractères. Une fois l'adresse mémoire récupérée par *mmap*, affichez son contenu à l'écran. Modifiez un des caractères dans la mémoire. Que se passe-t-il au niveau du fichier ?

Listing 4 – Solution de l'exercice 7

```
#include <sys/mman.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>


int main()
{
  char * adr;
  int fd = open("test.dat", O_RDONLY | 0666);

  /* Projette le segment de mémoire vide de taille 1024 avec
        accès en lecture/écriture */
  adr=mmap(NULL,1024,PROT_READ|PROT_WRITE,MAP_SHARED,fd,(
        off_t)0);

  if (adr==MAP_FAILED){perror("mmap");exit(2);}

  printf("Projection réussie à l'adresse %p\n",adr);

  while (1)
  {
    printf("%s", adr);
    sleep(2);
    adr[1] = 'e';
  }

  return 0;
}
```