Prácticas de Sistemas operativos

David Arroyo Guardeño

Escuela Politécnica Superior de la Universidad Autónoma de Madrid

Sexta semana: memoria compartida

Segunda práctica

1. Memoria compartida: ejercicio 2
2. Semáforos: ejercicios 4 y 5
3. Ejercicio final: ejercicios 6 y 7 (opcional)

Jueves 9 abril

Biblioteca shm I

1. Obtener una *llave*: **ftok**
2. Usar la llave para crear memoria compartida: **shmget** $\rightarrow$ **man shmget**
   - ✗ Entradas
     - $\Rightarrow$ Llave
     - $\Rightarrow$ Tamaño
     - $\Rightarrow$ Bandera: **IPC_CREAT** | **IPC_EXCL** | **mod_flags**
3. Asociar memoria local (asignación de dirección a puntero) a memoria compartida: **shmat**

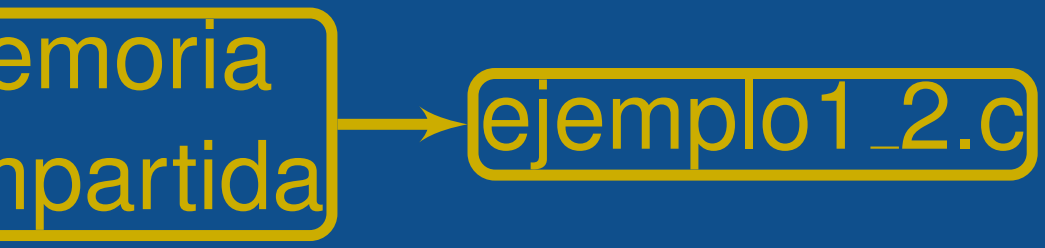
Biblioteca shm II

4. Desvincular del proceso memoria compartida: **shmdt**
5. Borrar memoria compartida: **shmctl**

```c
#define _XOPEN_SOURCE
#include <sys/types.h>
#include <sys/ipc.h>
#include <stdio.h>

main ()
{
  key_t key;
  char i = ' ';
  printf("Tamanyo key_t = %ld\n",sizeof(key));
  for(i='a';i<='d';i++){
    key = ftok(".",i);
    printf("proyecto= %c, clave = [%#x], MSB=%c\n",i,key,key>>24);
  }
  return 0;
}
```

Ejemplo1



```
...
#define FILEKEY "/bin/cat"
#define KEY 1300
#define MAXBUF 100

int main(int argc, char * argv[])
{
  int *buffer;  int key,id_zone;   int i;
  char c;
  key = ftok(FILEKEY,KEY);
  if(key==-1){
    fprintf(stderr,"Error with key\n");
    return -1;
  }

  id_zone = shmget(key,sizeof(int)*MAXBUF,IPC_CREAT|IPC_EXCL|SHM_R|SHM_W);

  if(id_zone == -1){
    fprintf(stderr,"Error with id_zone\n");
    return -1;
  }
  printf("ID zone shared memory:%i\n",id_zone);
  buffer = shmat(id_zone,(char *)0,0);
  if(buffer==NULL){
    fprintf(stderr,"Error reserve shared memory\n");
    return -1;
  }
  printf("Pointer buffer shared memory: %p\n",buffer);
  for(i=0;i<MAXBUF;i++){
    buffer[i] = i;
  }
  c = getchar();
  shmdt((char *)buffer);
  shmctl(id_zone,IPC_RMID,(struct shmid_ds *)NULL);
  return 0;
}
```

```
...
int main(int argc, char * argv[])
{
  int *buffer;
  int key,id_zone;
  int i;

  char c;
  key = ftok(FILEKEY,KEY);
  if(key==-1){
    fprintf(stderr,"Error with key\n");
    return -1;
  }

  id_zone = shmget(key,sizeof(int)*MAXBUF,SHM_R|SHM_W);

  if(id_zone == -1){
    fprintf(stderr,"Error with id_zone\n");
    return -1;
  }
  printf("ID zone shared memory:%i\n",id_zone);
  buffer = shmat(id_zone,(char *)0,0);
  if(buffer==NULL){
    fprintf(stderr,"Error reserve shared memory\n");
    return -1;
  }
  printf("Pointer buffer shared memory: %p\n",buffer);
  for(i=0;i<MAXBUF;i++){
    printf("%i\n",buffer[i]);
  }
  return 0;

}
```

Ejecución

- ✓ Ejecutar ejemplo1
- ✓ Hacer Ctrl+C
- ✓ ¿Qué ocurre?

Facilidades IPC

- ✓ **ipcs** $\rightarrow$ **man ipcs**
- ✓ En el ejemplo anterior
  - ✗ Hacer **ipcs −m**
  - ✗ Obtener el valor de la segunda columna para nuestro proceso: **shmid**
- ✓ **ipcrm** $\rightarrow$ **man ipcrm**
  - ✗ Una vez conocido el shmid de la memoria compartida que no hemos liberado
    - $\Rightarrow$ Hacer **ipcrm −m <shmid>**

Ejemplo 2:servidor

```
#define SHMSZ     27
main()
{
    char c;
    int shmid;
    key_t key;
    char *shm, *s;
    key = 5678;

    if ((shmid = shmget(key, SHMSZ, IPC_CREAT | 0666)) < 0) {
        perror("shmget");
        exit(1);
    }

    if ((shm = shmat(shmid, NULL, 0)) == (char *) -1) {
        perror("shmat");
        exit(1);
    }

    s = shm;

    for (c = 'a'; c <= 'z'; c++)
        *s++ = c;
    *s = 0;

    while (*shm != '*')
        sleep(1);

    exit(0);
}
```

Ejemplo: cliente

```c
#define SHMSZ     27

main()
{
    int shmid;
    key_t key;
    char *shm, *s;


    key = 5678;


    if ((shmid = shmget(key, SHMSZ, 0666)) < 0) {
        perror("shmget");
        exit(1);
    }


    if ((shm = shmat(shmid, NULL, 0)) == (char *) -1) {
        perror("shmat");
        exit(1);
    }


    for (s = shm; *s != 0; s++)
        putchar(*s);
    putchar('\n');


    *shm = '*';

    exit(0);
}
```

Ejemplo 3: cabecera

```c
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define ARRAY_SIZE    4000
#define MALLOC_SIZE  10000
#define SHM_SIZE     10000
#define SHM_MODE     (SHM_R | SHM_W)

#define TRUE         1
#define FALSE        0

#ifdef RUTINAS_SHMEM
static int shmid;
char *memoria;
#else
extern char *memoria;
#endif

int crea_mem() ;
int elimina_mem() ;
```

Ejemplo 3: implementación de biblioteca

```c
#define RUTINAS_SHMEM

#include "rutinas_shmem.h"


int crea_mem() {

    if ((shmid=shmget(IPC_PRIVATE, SHM_SIZE, SHM_MODE))<0) {
      fprintf(stderr, "error shmget(): %s, linea %d\n", __FILE__,__LINE__);
    } else if ((memoria=shmat(shmid, (char *)0, 0)) == (void *) -1) {
      fprintf(stderr, "error shmat(): %s, linea %d\n", __FILE__,__LINE__);
    } else {
       return TRUE;
    }
    return FALSE;
}


int elimina_mem() {

    if (shmctl(shmid, IPC_RMID, 0) < 0) {
      fprintf(stderr,"error shmctl(): %s, linea %d\n", __FILE__,__LINE__);
       return FALSE ;
    } else
       return TRUE ;
}
```

Ejemplo 3: programa ejemplo

```
#include "rutinas_shmem.h"

void modifica(int * m, int k) {
   int i;
   i=*m;
   sleep(rand()/RAND_MAX*5);
   i=i+k;
   sleep(rand()/RAND_MAX*5);;
   *m=i;
}
int main() {
   int  * num;   char * fin;    srand(time(NULL));

   if (!crea_mem())      fprintf(stderr, "error en crea_mem\n");

   num = (int *) memoria ;   fin = memoria + sizeof(int) ;    *num =  0 ;   *fin = 'p' ;
   if (fork() != 0) {
      int i;
      for (i=0; i< 100; i++)
         modifica(num, -10);
      while (*fin != 'x') ;
      printf("El valor original era 0; ahora es %d\n", *num);
      if (!elimina_mem())
         fprintf(stderr, "error en elimina_mem\n");
      exit(EXIT_SUCCESS);
   } else {
      int i;
      for (i=0; i< 100; i++)
         modifica(num, 10);
      *fin = 'x';
      exit(EXIT_SUCCESS);
   }
}
```

Referencias

- Francisco M. Márquez. Unix, Programación Avanzada. Editorial: Ra-Ma. 3ª Edición. ISBN: 84-7897-603-5