Sistemas Operativos
UTN Delta
Contenidos de la clase del lunes 5/11/2012.

Tema:
Implementación de la Problemática del Productor Consumidor utilizando memoria compartida y semáforos en el entorno operativo Linux.

Bibliografía:
Fundamentos de Sistemas Operativos. 7ma Edición. Silberschatz, Galvin, Gagne.
Capítulo 6 Sincronización de procesos.

Problema:
Se plantea el problema del productor consumidor con un buffer circular como mecanismo de comunicación con un contador de elementos que están en el buffer para detectar el buffer lleno en el productor y el buffer vacío en el consumidor y un semáforo de exclusión mutua como mecanismo de sincronización de acceso a la sección crítica.

6.1 Fundamentos

En el Capítulo 3 hemos desarrollado un modelo de sistema formado por procesos o hebras secuenciales cooperativas, los cuales se ejecutan de manera asíncrona y posiblemente compartiendo datos. Ilustramos este modelo con el problema del productor-consumidor, que es representativo de los sistemas operativos. Específicamente, en la Sección 3.4.1 hemos descrito cómo podría utilizarse un búfer limitado para permitir a los procesos compartir la memoria.

Consideremos de nuevo el búfer limitado. Como ya apuntamos, nuestra solución permite que haya como máximo BUFFER_SIZE −1 elementos en el búfer al mismo tiempo. Suponga que deseamos modificar el algoritmo para remediar esta deficiencia. Una posibilidad consiste en añadir una variable entera `counter`, inicializada con el valor 0. `counter` se incrementa cada vez que se añade un nuevo elemento al búfer y se decrementa cada vez que se elimina un elemento del búfer. El código para el proceso productor se puede modificar del siguiente modo:

```
while (true)
{
    /* produce un elemento en nextProduced */
    while (counter == BUFFER_SIZE)
        ; /* no hacer nada */
    buffer[in] = nextProduced;
    in = (in + 1) % BUFFER_SIZE;
```

171

172 Capítulo 6 Sincronización de procesos

```
    counter++;
}
```

El código para el proceso consumidor se puede modificar del modo siguiente

```
while (true)
{
    while (counter == 0)
        ; /* no hacer nada */
    nextConsumed = buffer[out];
    out = (out + 1) % BUFFER_SIZE;
    counter--;
    /* consume el elemento que hay en nextConsumed */
}
```

Objetivos:

1) Crear los segmentos de memoria compartida.
2) Crear el semáforo de exclusión mutua.
3) Inicializar los segmentos de memoria compartida.
4) Inicializar el semáforo.
5) Visualizar la inicialización del semáforo.
6) Configurar las operaciones wait y signal sobre el semáforo, en un archivo cabecera.
7) Escribir el programa productor.
8) Escribir el programa consumidor.
9) Script básico de compilación y visualización de recursos.

Desarrollo: Se pide programar en lenguaje C en el entorno operativo Linux, cada uno de los objetivos.

1) Crear los segmentos de memoria compartida.

```
//creashm.c
//Crea Memoria Compartida

#include <sys/types.h>
#include <sys/shm.h>
#include <sys/ipc.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>

#define SIZE_BUF 5

main()
{
  int shmid ;
//crea memoria compartida para el buffer circular
  shmid = shmget(0xa,SIZE_BUF,IPC_CREAT|IPC_EXCL|0600);
  printf("shmid = %d\n",shmid);
//si visualiza en pantalla shmid = -1 el shmget no tuvo exito

//crea memoria compartida para el guardar la cantidad de
//producciones
  shmid = shmget(0xb,sizeof(int),IPC_CREAT|IPC_EXCL|0600);
  printf("shmid = %d\n",shmid);
//si visualiza en pantalla shmid = -1 el shmget no tuvo exito
  exit(0);
}
```

2) Crear el semáforo de exclusión mutua.

```
//creasem.c
//Crea Semaforo de Exclusion Mutua

#include <sys/types.h>
#include <sys/sem.h>
#include <sys/ipc.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>

main()
{
```

```
  int semid ;
  //crea semaforo de exclusion mutua
  semid = semget(0xa,1,IPC_CREAT|IPC_EXCL|0600);
  printf("semid = %d\n",semid);
  //si visualiza en pantalla semid = -1 el semget no tuvo exito
  exit(0);
}
```

3) Inicializar los segmentos de memoria compartida.

```
//inicializashm.c
//Inicializa Memoria Compartida

#include <sys/types.h>
#include <sys/shm.h>
#include <sys/ipc.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

#define SIZE_BUF 5

main()
{
  int shmid ;
//Obtiene el shmid de la memoria compartida para el buffer circular
  shmid = shmget(0xa,0,0);
  printf("shmid = %d\n",shmid);
//si visualiza en pantalla shmid = -1 el shmget no pudo obtener el
//shmid
  if (shmid != -1)
   {
     char * dirbuf = (char *) shmat(shmid,0,0); //mapea la
direccion //del shared memory en dirbuf
     if (dirbuf != NULL)
      {
        memset(dirbuf,'_',SIZE_BUF);
        shmdt(dirbuf);
      }
   }
//Obtiene el shmid de la  memoria compartida para guardar la
//cantidad de producciones
  shmid = shmget(0xb,0,0);
  printf("shmid = %d\n",shmid);
//si visualiza en pantalla shmid = -1 el shmget no pudo obtener el
//shmid
  if (shmid != -1)
   {
     int * dircon = (int *) shmat(shmid,0,0);  //mapea la
direccion //del shared memory en dircon
     if (dircon != NULL)
      {
        *dircon = 0;
        shmdt(dircon);
      }
   }
   exit(0);
}
```

4) Inicializar el semáforo.

```
//inicializasem.c
//Inicializa Semaforo de Exclusion Mutua

#include <sys/types.h>
#include <sys/sem.h>
#include <sys/ipc.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>

main()
{
  int semid ;

//Obtiene el semid del Semaforo de Exclusion Mutua
  semid = semget(0xa,0,0);
  printf("semid = %d\n",semid);
//si visualiza en pantalla semid = -1 el semget no pudo obtener el
//semid
  if (semid != -1)
     semctl(semid,0,SETVAL,1); //inicializa el semaforo en 1

  exit(0);
}
```

5) Visualizar la inicialización del semáforo.

```
//versem.c
//Visualiza el valor del Semaforo de Exclusion Mutua

#include <sys/types.h>
#include <sys/sem.h>
#include <sys/ipc.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>

main()
{
  int semid ;

//Obtiene el semid delSemaforo de Exclusion Mutua
  semid = semget(0xa,0,0);
  printf("semid = %d\n",semid);
//si visualiza en pantalla semid = -1 el semget no pudo obtener el
//semid
  if (semid != -1)
     printf("Semaforo = %d\n",semctl(semid,0,GETVAL)); //muestra
el //valor de semaforo

  exit(0);
}
```

6) Configurar las operaciones wait y signal sobre el semáforo, en un archivo cabecera.

```
//operIPC.h
//Funciones que llaman a la operacion wait y signal

#include <sys/types.h>
#include <sys/sem.h>
#include <sys/ipc.h>
#include <unistd.h>
#include <stdlib.h>

//operacion wait: while(sem <=0);
//                    sem = sem - 1;
void P(int semid, int sem)
{
  struct sembuf buf;
  buf.sem_num = sem;
  buf.sem_op  = -1;
  buf.sem_flg = 0;
  semop(semid,&buf,1); //system call para la operacion atomica wait
}

//operacion signal: sem = sem + 1;
void V(int semid, int sem)
{
  struct sembuf buf;
  buf.sem_num = sem;
  buf.sem_op  = 1;
  buf.sem_flg = 0;
  semop(semid,&buf,1); //system call para la operacion atomica signal
}
```

7) Escribir el programa productor.

```
//productor.c
//Productor produce en por tiempo indefinido

#include "operIPC.h"
#include <sys/shm.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

#define SIZE_BUF 5

char produccion(void);

main()
{
  int shmidbuf, shmidcon, semid ;

//Obtiene el shmid de la memoria compartida para el buffer circular
  shmidbuf = shmget(0xa,0,0);
  printf("shmidbuf = %d\n",shmidbuf);
//si visualiza en pantalla shmidbuf = -1 el shmget no pudo obtener
el //shmidbuf

//Obtiene el shmidcon de la  memoria compartida para guardar la
//cantidad de producciones
  shmidcon = shmget(0xb,0,0);
  printf("shmidcon = %d\n",shmidcon);
```

```
//si visualiza en pantalla shmidcon = -1 el shmget no pudo obtener
el //shmidcon

//Obtiene el semid del semaforo
  semid = semget(0xa,0,0);
  printf("semid = %d\n",semid);
//si visualiza en pantalla semid = -1 el semget no pudo obtener el
//semid

  if (shmidbuf != -1 && shmidcon != -1 && semid != -1)
   {
     char * dirbuf = (char *) shmat(shmidbuf,0,0); //mapea la
//direccion del shared memory shmidbuf en dirbuf
     int * dircon = (int *) shmat(shmidcon,0,0);  //mapea la direccion
//del shared memory shmidcon en dircon
     if (dirbuf != NULL && dircon != NULL)
      {
        int ent = 0 ;
        while(1)
        {
           while(*dircon == SIZE_BUF); //Buffer lleno, no produce,
//espera activa
           P(semid,0);
           dirbuf[ent] = produccion();
           (*dircon)++;
           V(semid,0);
           ent = (ent + 1) % SIZE_BUF ;
        }
      }
   }
   exit(0);
}

char produccion()
{
  static int letra = 'A' - 1;
  if (letra == 'Z')
    letra = 'A';
  else
    letra++ ;
  return letra;
}
```

8) Escribir el programa productor.

```
//consumidor.c
//Consumidor consume por tiempo indefinido

#include "operIPC.h"
#include <sys/shm.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

#define SIZE_BUF 5


main()
{
  int shmidbuf, shmidcon, semid ;

//Obtiene el shmid de la memoria compartida para el buffer circular
```

```
  shmidbuf = shmget(0xa,0,0);
  printf("shmidbuf = %d\n",shmidbuf);
//si visualiza en pantalla shmidbuf = -1 el shmget no pudo obtener
el //shmidbuf

//Obtiene el shmidcon de la  memoria compartida para guardar la
//cantidad de producciones
  shmidcon = shmget(0xb,0,0);
  printf("shmidcon = %d\n",shmidcon);
//si visualiza en pantalla shmidcon = -1 el shmget no pudo obtener
el //shmidcon

//Obtiene el semid del semaforo
  semid = semget(0xa,0,0);
  printf("semid = %d\n",semid);
//si visualiza en pantalla semid = -1 el semget no pudo obtener el
//semid

  if (shmidbuf != -1 && shmidcon != -1 && semid != -1)
   {
     char * dirbuf = (char *) shmat(shmidbuf,0,0); //mapea la
//direccion del shared memory shmidbuf en dirbuf
     int * dircon = (int *) shmat(shmidcon,0,0);  //mapea la direccion
//del shared memory shmidcon en dircon
     if (dirbuf != NULL && dircon != NULL)
      {
        int sal = 0 ;
        while(1)
        {
           while(*dircon == 0); // buffer vacio, no consume,
espera //activa
           P(semid,0);
           printf("letra consumida %c\n",dirbuf[sal]);
           (*dircon)--;
           V(semid,0);
           sal = (sal + 1) % SIZE_BUF ;
           usleep(50000);
        }
      }
   }
   exit(0);
}
```

9) Script básico de compilación y visualización de reursos.

```
cc creasem.c -o creasem
cc creashm.c -o creashm
cc inicializashm.c -o inicializashm
cc inicializasem.c -o inicializasem
cc versem.c -o versem
cc productor.c -o productor
cc consumidor.c -o consumidor
./creashm
./creasem
ipcs
./inicializashm
./inicializasem
./versem
```