INSTITUTO POLITÉCNICO NACIONAL ESCUELA SUPERIOR DE CÓMPUTO

INGENIERÍA EN SISTEMAS COMPUTACIONALES

SISTEMAS OPERATIVOS

Práctica 3

Comunicación entre procesos / hilos

- Alumnos:
  1) Arcos Hermida Aldo Alejandro (N° lista 5)
  2) Chávez Becerra Bianca Cristina (N° lista 9)
  3) Islas Osorio Enrique (N° lista 20)
  4) Juárez Cabrera Jessica (N° lista 21)
  5) Palmerin García Diego (N° lista 28)
- Grupo: 4CM1
- Materia: Sistemas operativos
- Profesor: Araujo Díaz David

Objetivo

Utilizar el mecanismo de memoria compartida para comunicar dos o más procesos en un sistema operativo LINUX/UNIX.

Descripción

La memoria compartida (Shared Memory) es una forma implícita de comunicación entre procesos (IPC). Es, por tanto, un método altamente eficiente para compartir información entre procesos.

Los procesos pueden comunicarse directamente entre sí compartiendo partes de su espacio de direccionamiento virtual, por lo que podrán leer y/o escribir datos en la memoria compartida. Para conseguirlo, se crea una región o segmento fuera del espacio de direccionamiento de un proceso y cada proceso que necesite acceder a dicha región, la incluirá como parte de su espacio de direccionamiento virtual. El sistema permite que existan varias regiones compartibles, cada una compartida por un subconjunto de procesos. Además, cada proceso puede acceder a varias regiones.

Posix proporciona dos formas de compartir áreas de memoria entre procesos no relacionados:

(1) Archivo mapeado en memoria: primero abra la función open y luego llame a la función mmap para mapear el descriptor obtenido a un archivo en el espacio de direcciones del proceso actual (el que se usó en la nota anterior).

(2) Objeto de área de memoria compartida: primero shm_open abre un nombre de IPC Posix (o un nombre de ruta en el sistema de archivos), y luego llama a mmap para asignar el descriptor devuelto al espacio de direcciones del proceso actual.

Los dos métodos en su mayoría necesitan llamar a mmap, la diferencia radica en el método de obtener el descriptor como uno de los parámetros de mmap.

2. Objetos del área de memoria compartida de Posix

El área de memoria compartida de Posix implica los siguientes requisitos de dos pasos:

(1) Especifique un parámetro de nombre para llamar a shm_open para crear un nuevo objeto de área de memoria compartida o abrir un objeto de área de memoria compartida existente.

(2) Llame a mmap para asignar esta área de memoria compartida al espacio de direcciones del proceso de llamada.

Nota: mmap se usa para mapear un objeto de área de memoria al espacio de direcciones del proceso de llamada es un descriptor abierto del objeto.

Se programarán en C tres programas :

- El primero creará un objeto de memoria compartida

- El segundo escribirá un texto en memoria compartida

- El tercero leerá de ese objeto de memoria compartida.

Se escribirá, compilará y ejecutará el código de cada programa.

Código fuente

```
/*
   Crea un objeto de memoria "/myMemoryObj"
   El objeto se puede encontrar: /dev/shm
   para ser compilado con "-lrt"
*/
#include <stdio.h>
#include <stdlib.h>
#include <sys/mman.h>
#include <sys/stat.h>        /* para modo de contantes */
#include <fcntl.h>           /* para O_* constantes */
#include <unistd.h>
#include <sys/types.h>

#define SIZEOF_SMOBJ 200
#define SMOBJ_NAME  "/myMemoryObj"

int main(void)
{
   int fd;
   fd = shm_open (SMOBJ_NAME, O_CREAT | O_RDWR  , 00700); /* crear s.m objeto*/
   if(fd == -1)
   {
       printf("Error descritor de archivo \n");
       exit(1);
   }
   if(-1 == ftruncate(fd, SIZEOF_SMOBJ))
   {
       printf("Error memoria compartida no puede ser redimensionada \n");
       exit(1);
   }

   close(fd);

   return 0;
}
```

```c
/*

Escribe en un objeto de memoria creado previamente "/myMemoryObj"
   para ser compilado con "-lrt"
*/

#include <stdio.h>
#include <stdlib.h>
#include <sys/mman.h>
#include <sys/stat.h>        /* For mode constants */
#include <fcntl.h>           /* For O_* constants */
#include <unistd.h>
#include <sys/types.h>
#include <errno.h>
#include <string.h>

#define SMOBJ_NAME  "/myMemoryObj"

int main(void)
{
   int fd;
   char buf[] = "Hola, este es el proceso de escritura";
   char *ptr;

   fd = shm_open (SMOBJ_NAME,  O_RDWR  , 00200); /* open s.m object*/
   if(fd == -1)
   {
       printf("Error file descriptor %s\n", strerror(errno));
       exit(1);
   }

   ptr = mmap(NULL, sizeof(buf), PROT_WRITE, MAP_SHARED, fd, 0);
   if(ptr == MAP_FAILED)
   {
      printf("Map failed in write process: %s\n", strerror(errno));
      exit(1);
   }

   memcpy(ptr,buf, sizeof(buf));
   printf("%d \n", (int)sizeof(buf));
   close(fd);

   return 0;
}
```

```c
/*

Lee de un objeto de memoria creado previamente "/myMemoryObj"
   para ser compilado con "-lrt"
*/

#include <stdio.h>
#include <stdlib.h>
#include <sys/mman.h>
#include <sys/stat.h>        /* For mode constants */
#include <fcntl.h>           /* For O_* constants */
#include <unistd.h>
#include <sys/types.h>
#include <errno.h>
#include <string.h>

#define SMOBJ_NAME  "/myMemoryObj"

int main(void)
{
   int fd;
   char *ptr;
   struct stat shmobj_st;

   fd = shm_open (SMOBJ_NAME,  O_RDONLY  , 00400); /* open s.m object*/
   if(fd == -1)
   {
       printf("Error file descriptor %s\n", strerror(errno));
       exit(1);
   }

   if(fstat(fd, &shmobj_st) == -1)
   {
       printf(" error fstat \n");
       exit(1);
   }
   ptr = mmap(NULL, shmobj_st.st_size, PROT_READ, MAP_SHARED, fd, 0);
   if(ptr == MAP_FAILED)
   {
      printf("Map failed in read process: %s\n", strerror(errno));
      exit(1);
   }

   printf("%s \n", ptr);
   close(fd);

   return 0;
}
```

Creando el objeto de memoria

```
enri@enri-VirtualBox:~$ cd Desktop
enri@enri-VirtualBox:~/Desktop$ gcc crear.c -o crear -lrtS
```

Mostrando el objeto de memoria, se crearon dos con nombre diferente

```
enri@enri-VirtualBox:~$ cd /dev/shm
enri@enri-VirtualBox:/dev/shm$ ls -l
total 0
-rwx------ 1 enri enri 200 feb 27 13:06 myMemoryObj
enri@enri-VirtualBox:/dev/shm$ ls -l
total 0
-rwx------ 1 enri enri 200 feb 27 13:08 miObjetoDeMemoria
-rwx------ 1 enri enri 200 feb 27 13:06 myMemoryObj
enri@enri-VirtualBox:/dev/shm$ S
```

Mostrando el proceso de escritura en myMemoryObj

```
-rwx------ 1 enri enri 200 feb 27 13:08 miObjetoDeMemoria
-rwx------ 1 enri enri 200 feb 27 13:20 myMemoryObj
enri@enri-VirtualBox:/dev/shm$ cat myMemoryObj
Hola, este es el proceso de escrituraenri@enri-VirtualBox:/dev/shm$ S
```

Compilación y ejecución del proceso de lectura que nos regresa lo que habíamos ingresado en el proceso de escritura

```
enri@enri-VirtualBox:~/Desktop$ gcc lectura.c -o lectura -lrt
enri@enri-VirtualBox:~/Desktop$ ./lectura
Hola, este es el proceso de escritura
enri@enri-VirtualBox:~/Desktop$
```

Conclusiones

En la realización de la práctica se logró utilizar el mecanismo de memoria compartida para establecer una zona en común de memoria entre dos objetos que permita su comunicación. La memoria compartida es la tecnología de comunicación entre procesos más rápida, flexible y eficiente para compartir información entre procesos. La memoria que maneja un proceso, y también la compartida, va a ser virtual, por lo que su dirección física puede variar con el tiempo. Esto no va a plantear ningún problema, ya que los procesos no generan direcciones físicas, sino virtuales.

Referencias

- Área de memoria compartida Posix - programador clic. (s. f.). programerclick. Recuperado 27 de febrero de 2022, de https://programmerclick.com/article/73871146075/

- Daniel P. Bovet, Marco Cesati, “Understanding The Linux Kernel Understanding The Linux Kernel”. 3rd Edition 3rd Edition. O'Reilly Media, Inc., 2005. ISBN: 0-596-00565-2

- DISC, & Great Britain: Dept. of Trade & Industry. (1993). POSIX: A Technical Guide. Blackwell.