Programación Avanzada

Curso 2023/2024

prof. Claudio Rossi

Practica 3: comunicación a través de colas de mensajes

Las colas de mensajes representan un método mediante el cual los procesos pueden intercambiarse o pasarse datos (IPC). Una cola de mensajes puede ser creada por un proceso y utilizada por múltiples procesos que leen o escriben datos en la cola. La cola de mensajes es administrada por el sistema operativo o el *kernel*. Los programas de aplicación (o sus procesos) crean colas de mensajes pudiendo enviar y recibir mensajes a través de un Programa de Interfaz de Aplicación (API).

Objetivos:

1. Uso básico de colas de mensajes
2. Problema propuesto

Comandos de S.O.:

- `ipcs`: imprime en pantalla información del estado de las primitivas de IPC (colas, semáforos, memoria compartida)
- `ipcrm`: elimina estructuras de IPC (*ejemplo:* `ipcrm -q 123456` *elimina la cola de mensajes con id. 123456*)

Estucturas de datos utilizadas:

- *struct msgbuf ,* presenta dos campos: *long mtype*, permite definir al usuario una etiqueta asociada al mensaje. *char mtext[size]*, texto del mensaje que se envía:

```
struct msgbuf{
    long mtype;
    char mtext[size]; // size es un numero ≥ 0
  }
```

Funciones utilizadas:

- int *msgget(int key, int flag)*: proporciona un identificador para la cola de clave *key*, con permisos *flag* (máscara de bits con instrucciones para la creación y permisos de acceso con formato rwxrwxrwx)

- int *msgsnd(int id, struct msgbuf * buf, int size, int flag)*: envía el mensaje almacenado en el segundo campo de *buf*, de tamaño *size*, a la cola con identificador *id*. El cuarto parámetro sirve para controlar el comportamiento según el nivel de ocupación de la cola (p.e. bloqueante o no).

- int *msgrcv(int id, struct msgbuf * buf, int size, long type, int flag)*: recupera el primer mensaje con identificador *type*, de tamaño *size* almacenado en la cola con identificador *id*, en el segundo campo de *buf*. El último parámetro sirve para

controlar el comportamiento según el nivel de ocupación de la cola (p.e. bloqueante o no).

- int *msgctl(int id, int cmd, struct msqid_ds * buf)*: proporciona diversos modos de control sobre la cola de mensaje *id*. La operación queda definida por *cmd*: IPC_RMID, borra la cola. El tercer parámetro es una estructura con diversos campos tales como, permisos, número de mensajes encolados, etc.

- int *sprintf(char *buffer, const char *formato,...)*: su funcionamiento es idéntico a la función *printf*, salvo que la salida generada se pone en el array apuntado por *buffer*. El valor devuelto es igual al número de caracteres puesto en el array.

- void *perror*(const char *message): imprime un mensaje de error asociado a la cadena que tiene como argumento.

- int *atoi(const char *cad)*: convierte la cadena de caracteres que apunta *cad* a un valor int. La cadena debe contener un número válido de entero.

Parte 1.
Se escribirán dos programas. El primero crea una cola de mensajes (*msgget*) y escribe un mensaje en la cola (*msgsnd*). El segundo se conecta a la cola creada (*msgget*) y espera recibir un mensaje (*msgrcv*) y cierra la cola (*msgctl*). Para usar la misma cola, los dos programas deberá compartir la misma "clave" de la cola.

Comprobad, a través del comando del S.O. *ipcs*, la creación de la cola !

Programa 1

```
/* Programa de ejemplo del uso de colas de mensajes -- productor*/

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

/* Definición de la clave de la cola */
#define Clave_cola 1


/* Función auxiliar que crea la cola, retornando el identificador */
int crearcola(int key)
{
  int msgqid;

  msgqid = msgget(key, IPC_CREAT|0666);
  switch(msgqid)
      {
      case -1:
        perror("msgget");
```

```
        return(-1);
      default:
        return msgqid;
      }
} /* Fin función auxiliar */


/* Función principal */
int main(void)
{
  int id_cola, i, ret;

  struct msgbuf{
      long mtype;
      char mtext[15];
  } mensaje;


  /* Llamada a la función de creación de colas */

  id_cola=crearcola(Clave_cola);

  if(id_cola==-1)
      {
        printf("No se ha podido crear la cola !\n");
        exit(EXIT_FAILURE);
      }

  /* Inicialización de los campos de la estructura */

  mensaje.mtype=1;

  for (i=0;i<10;i++)
      {
        sleep(1);
        sprintf(mensaje.mtext,"Mensaje %d",i);

        /* Envío del mensaje (introducción en la cola) */
        ret=msgsnd(id_cola,&mensaje,sizeof(mensaje.mtext),0);

        if(ret == -1)
            {
              perror("msgsnd");
              exit(EXIT_FAILURE);
            }

        printf("He enviado el mensaje: %s\n",mensaje.mtext);
      }
}
```

Programa 2

```
/* Programa de ejemplo del uso de colas de mensajes -- consumidor*/

#include <sys/ipc.h>
#include <sys/msg.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
```

```
/* Definición de la clave de la cola */
#define Clave_cola 1


/* Función auxiliar que crea la cola, retornando el identificador */
int crearcola(int key)
{
  int msgqid;

  msgqid = msgget(key, IPC_CREAT|0666);
  switch(msgqid)
      {
      case -1:
        perror("msgget");
        return(-1);
      default:
        return msgqid;
      }
} /* Fin función auxiliar */


/* Función principal */
int main(void)
{
  int id_cola, i, ret;


  struct msgbuf{
      long mtype;
      char mtext[15];
  } mensaje;


  /* Llamada a la función de creación de colas */

  id_cola=crearcola(Clave_cola);

  if(id_cola==-1)
      {
        printf("No se ha podido crear la cola !\n");
        exit(EXIT_FAILURE);
      }

  /* Inicialización de los campos de la estructura */

  mensaje.mtype=1;

  for (i=0;i<10;i++)
      {

        /* Lectura de datos de la cola */
        ret=msgrcv(id_cola,&mensaje,sizeof(mensaje.mtext),1,0);

        if(ret == -1)
            {
              perror("msgrcv");
              exit(EXIT_FAILURE);
            }

        printf("He recibido el mensaje: %s\n",mensaje.mtext);
```

```
      }

  // elimina la cola del sistema
  ret=msgctl(id_cola,IPC_RMID,(struct msqid_ds*)0);

  if(ret==-1)
      perror("msgctl");
  else
      printf("La cola se ha cerrado correctamente\n");

}
```

Parte 2.

Basándose en los programas de ejemplo proporcionados, y en la practica anterior, escribir otro para que se cree una cola de mensajes para comunicación entre un proceso padre y un hijo. El Padre deberá adivinar un numero elegido por el hijo (entre 0 y 100). Para ello, enviará sus intentos al hijo a través de mensajes. El hijo contestará con un mensaje para decir si el numero es mayor, menor o si ha acertado. En caso de acierto los proceso terminan. Usar el campo "*mtype*" de la estructura *msgbuf* para identificar los mensajes (ej: el padre envía mensajes de tipo 1 y el hijo de tipo 2)

(Como en la práctica anterior, para convertir un numero en cadena de caracteres usar la función *sprintf*, para convertir una cadena de caracteres en un numero usar la función *atoi.*)

Ejemplo:

```
char buffer[100];
char msg[100];
int i,x;

sprintf(msg,"%d",x);  // transforma el numero 10 en la cadena "10"
intento=atoi(buffer); // transforma la cadena en numero (ej: "123"->123)
```

Para generar un numero al azar:

```
srandom(getpid());   // inicializa la semilla para los numeros
x=(int)(100.0*(random()/(float)RAND_MAX));  // x estará entre 0 y 100
```