

```

#include <stdio.h>
#include <string.h>
#include <sys/msg.h>
#include <errno.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

//
// Estructura para los mensajes que se quieren enviar y/o recibir. Deben llevar
// obligatoriamente como primer campo un long para indicar un identificador
// del mensaje.
// Los siguientes campos son la información que se quiera transmitir en el
// mensaje. Cuando más adelante, en el código, hagamos un cast a
// (struct msgbuf *), todos los campos de datos los verá el sistema como
// un único (char *)
//
typedef struct
{
    long Id_Mensaje;
    int Dato_Numerico;
    char Mensaje[10];
} Mi_Tipo_Mensaje;

int main()
{
    key_t Clave1;
    int Id_Cola_Mensajes;
    Mi_Tipo_Mensaje Un_Mensaje;
    printf("Soy el proceso %d", (int)getpid());
    //
    // Igual que en cualquier recurso compartido (memoria compartida, semaforos
    // o colas) se obtien una clave a partir de un fichero existente cualquiera
    // y de un entero cualquiera. Todos los procesos que quieran compartir este
    // semaforo, deben usar el mismo fichero y el mismo entero.
    //
    Clave1 = ftok ("/bin/ls", 33);
    if (Clave1 == (key_t)-1)
    {
        printf("Error al obtener clave para cola mensajes");
        exit(-1);
    }

    //
    // Se crea la cola de mensajes y se obtiene un identificador para ella.
    // El IPC_CREAT indica que cree la cola de mensajes si no lo está ya.
    // el 0600 son permisos de lectura y escritura para el usuario que lance
    // los procesos. Es importante el 0 delante para que se interprete en
    // octal.
    //
    Id_Cola_Mensajes = msgget (Clave1, 0600 | IPC_CREAT);
    if (Id_Cola_Mensajes == -1)
    {
        printf("Error al obtener identificador para cola mensajes");
        exit (-1);
    }
    else{
        printf( "Cola %d\n",Id_Cola_Mensajes);
    }

    //
    // Se recibe un mensaje del otro proceso. Los parámetros son:
    // - Id de la cola de mensajes.
    // - Dirección del sitio en el que queremos recibir el mensaje,
    // convirtiéndolo en puntero a (struct msgbuf *).
    // - Tamaño máximo de nuestros campos de datos.
    // - Identificador del tipo de mensaje que queremos recibir. En este caso
    // se quiere un mensaje de tipo 1, que es el que envia el proceso cola1.cc
    // - flags. En este caso se quiere que el programa quede bloqueado hasta
    // que llegue un mensaje de tipo 1. Si se pone IPC_NOWAIT, se devolvería
    // un error en caso de que no haya mensaje de tipo 1 y el programa
    // continuaría ejecutándose.
    //
    msgrcv (Id_Cola_Mensajes, (struct msgbuf *)&Un_Mensaje,
        sizeof(Un_Mensaje.Dato_Numerico) + sizeof(Un_Mensaje.Mensaje),
        3, 0);

    printf("Recibido mensaje tipo 3\n");
    printf("Dato_Numerico = %d\n",Un_Mensaje.Dato_Numerico );
    printf("Mensaje = %s\n",Un_Mensaje.Mensaje );

    //
    // Se rellenan los campos del mensaje que se quiere enviar.
    // El Id_Mensaje es un identificador del tipo de mensaje. Luego se podrá
    // recoger aquellos mensajes de tipo 1, de tipo 2, etc.
    // Dato_Numerico es un dato que se quiera pasar al otro proceso. Se pone,
    // por ejemplo 13.
    // Mensaje es un texto que se quiera pasar al otro proceso.
    //
    Un_Mensaje.Id_Mensaje = 2;
    Un_Mensaje.Dato_Numerico = 13;
    strcpy (Un_Mensaje.Mensaje, "Adios");

    //
    // Se envia el mensaje. Los parámetros son:
    // - Id de la cola de mensajes.
    // - Dirección al mensaje, convirtiéndola en puntero a (struct msgbuf *)
    // - Tamaño total de los campos de datos de nuestro mensaje, es decir
    // de Dato_Numerico y de Mensaje
    // - Unos flags. IPC_NOWAIT indica que si el mensaje no se puede enviar
    // (habitualmente porque la cola de mensajes esta llena), que no espere
    // y de un error. Si no se pone este flag, el programa queda bloqueado
    // hasta que se pueda enviar el mensaje.

    sleep(10);
    //

    return 0;
}

```