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


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[3];
 Mi_Tipo_Mensaje Resp;

 printf("Soy el proceso %d\n", (int)getpid());
 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 rellenan los campos del mensaje que se quiere enviar.
 //
 Un_Mensaje[0].Id_Mensaje = 1;
 Un_Mensaje[0].Dato_Numerico = 29;
 strcpy (Un_Mensaje[0].Mensaje, "Hola");

 Un_Mensaje[1].Id_Mensaje = 3;
 Un_Mensaje[1].Dato_Numerico = 25;
 strcpy (Un_Mensaje[1].Mensaje, "Hola1");

 Un_Mensaje[2].Id_Mensaje = 5;
 Un_Mensaje[2].Dato_Numerico = 24;
 strcpy (Un_Mensaje[2].Mensaje, "Hola2");

 //
 //  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.
 //
 msgsnd (Id_Cola_Mensajes, (struct msgbuf *)&(Un_Mensaje[0]),
    sizeof(Un_Mensaje[0].Dato_Numerico)+sizeof(Un_Mensaje[0].Mensaje),
    IPC_NOWAIT);
 msgsnd (Id_Cola_Mensajes, (struct msgbuf *)&(Un_Mensaje[1]),
    sizeof(Un_Mensaje[1].Dato_Numerico)+sizeof(Un_Mensaje[1].Mensaje),
    IPC_NOWAIT);
 msgsnd (Id_Cola_Mensajes, (struct msgbuf *)&(Un_Mensaje[2]),
    sizeof(Un_Mensaje[2].Dato_Numerico)+sizeof(Un_Mensaje[2].Mensaje),
    IPC_NOWAIT);
printf("ya entregue el mensaje\n");
 //
 //  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 2. Si ponemos tipo 1, se extrae el mensaje
 //  que se acaba de enviar en la llamada anterior a msgsnd().
 //  - flags. En este caso se quiere que el programa quede bloqueado hasta
 //  que llegue un mensaje de tipo 2. Si se pone IPC_NOWAIT, se devolvería
 //  un error en caso de que no haya mensaje de tipo 2 y el programa
 //  continuaría ejecutándose.
 //
 msgrcv (Id_Cola_Mensajes, (struct msgbuf *)&Resp,
    sizeof(Resp.Dato_Numerico) + sizeof(Resp.Mensaje),
    2, 0);

 printf("Recibido mensaje tipo 2\n");
 printf("Dato_Numerico = %d\n",Resp.Dato_Numerico );
 printf("Mensaje = %s\n", Resp.Mensaje );
sleep(20);
 //
 //  Se borra y cierra la cola de mensajes.
 // IPC_RMID indica que se quiere borrar. El puntero del final son datos
 // que se quieran pasar para otros comandos. IPC_RMID no necesita datos,
 // así que se pasa un puntero a NULL.
 //
 msgctl (Id_Cola_Mensajes, IPC_RMID, (struct msqid_ds *)NULL);
 return 0;
}
```