

PRÁCTICA 6

Programación de colas de mensajes en C para Linux

Estudiante	Escuela	Asignatura
Rodrigo Infanzón Acosta rinfanzona@ulasalle.edu.pe	Carrera Profesional de Ingeniería de Software	Sistemas Operativos

Informe	Tema	Duración
07	Programación de colas de mensajes en C para Linux	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - B	15/11/24	22/11/24

Índice

1. Objetivos	1
2. Recursos y herramientas utilizados	1
3. Ejercicios propuestos	2
3.1. Ejercicio 1	2
3.2. Ejercicio 2	4
3.3. Ejercicio 3	7

1. Objetivos

- El alumno comprender a y analizar a el funcionamiento interno de los sistemas operativos desde la utilizaci on de comandos hasta la programaci on de procesos.
- El alumno deber a de probar, analizar y explicar el funcionamiento de los mecanismos de comunicaci on entre procesos utilizando las colas de mensajes.

2. Recursos y herramientas utilizados

- Sistema operativo utilizado: Windows 10 pro 22H2 de 64 bits SO. 19045.4170.
- Hardware: Ryzen 7 5700X 4.0 GHz, RAM 32 GB DDR4 3200 MHz, RTX 4060 Asus Dual.
- Virtual Box 7.0.20-163906-Win
- Git 2.44.0.
- Sistema invitado utilizado: Fedora Linux 40 Cinnamon Spin
- Conocimientos básicos en Git.

- Conocimientos básicos en sistemas operativos.
- Conocimientos básicos en Linux.
- C (gcc).

3. Ejercicios propuestos

3.1. Ejercicio 1

Analice y describa la actividad que realiza el siguiente código. Explique cómo se ha creado una cola de mensajes para permitir el envío de información, incluso después de haber finalizado la ejecución del programa:

```
1 #include <sys/types.h>
2 #include <sys/ipc.h>
3 #include <sys/msg.h>
4 #include <unistd.h>
5 #include <errno.h>
6 #include <stdio.h>
7 #include <stdlib.h>
8
9 #define CLAVE_MSG 2000
10 #define OK 0
11 #define ERROR -1
12 #define INFO 150
13 #define TIPO 7
14
15 typedef struct {
16     long tipo;
17     int info;
18 } MENSAJE;
19
20 int main() {
21     int qid, opcion;
22     MENSAJE msg, msg2;
23     int lector = 0;
24
25     // Cambiar SHM_R|SHM_W por permisos correctos
26     qid = msgget(CLAVE_MSG, IPC_CREAT | IPC_EXCL | 0666);
27     if (qid == ERROR) {
28         if (errno == EEXIST) {
29             printf("Ya existe una cola de mensajes, accediendo...\n");
30             qid = msgget(CLAVE_MSG, 0666);
31         } else {
32             perror("msgget: ");
33             exit(errno);
34         }
35     }
36     printf("Cola de mensajes creada...\n");
37
38     printf("Enviar mensaje[1] o leer mensaje[2]: ");
39     scanf("%d", &opcion);
40
41     switch(opcion) {
42         case 1:
```

```
43     msg.tipo = TIPO; // pid del destinatario
44     msg.info = INFO; // informacin a transmitir
45
46     printf("Enviando mensaje...\n");
47     if (msgsnd(qid, &msg, sizeof(MENSAJE) - sizeof(long), 0) == ERROR) {
48         perror("msgsnd: ");
49         exit(errno);
50     }
51     printf("Mensaje enviado...\n");
52     break;
53 case 2:
54     printf("Leyendo el primer mensaje de la cola...\n");
55     if (msgrcv(qid, &msg2, sizeof(MENSAJE) - sizeof(long), 0, 0) == ERROR) {
56         perror("msgrcv: ");
57         exit(errno);
58     }
59
60     printf("Mensaje recibido de tipo = %ld con info = %d\n", msg2.tipo, msg2.info);
61     lector = 1;
62     break;
63 default:
64     printf("No ha elegido ninguna opcin\n");
65 }
66
67 // Proceso de comunicacin finalizado
68 if (lector == 1) {
69     if (msgctl(qid, IPC_RMID, NULL) == ERROR) {
70         perror("msgctl: ");
71         exit(errno);
72     }
73     printf("Cola de mensajes eliminada\n");
74 }
75
76 exit(OK);
77 }
```

Análisis: el programa utiliza la API de colas de mensajes de System V para permitir la comunicación asincrónica entre procesos. Estas colas permiten enviar y recibir mensajes de manera persistente, incluso si el programa termina su ejecución.

Actividades que realiza:

1. **Creación de la cola:** se crea una cola con `msgget` usando la clave fija `CLAVE_MSG` y permisos `0666`. Si la cola ya existe, se accede a ella sin recrearla.
2. **Interacción del usuario:** el usuario elige entre:
 - **Enviar mensaje:** agrega un mensaje con tipo y valor definido.
 - **Leer mensaje:** recupera el primer mensaje almacenado (FIFO).
3. **Persistencia:** la cola permanece en el sistema incluso después de finalizar el programa, permitiendo que otros procesos accedan a los mensajes enviados.
4. **Eliminación de la cola:** si se procesan mensajes, la cola se elimina con `msgctl` para liberar recursos.

Persistencia de la cola de mensajes: el sistema mantiene la cola activa hasta que se elimine explícitamente con `msgctl`. Esto permite que otros programas puedan leer o escribir mensajes posteriormente, siempre que utilicen la misma clave y tengan los permisos adecuados.

3.2. Ejercicio 2

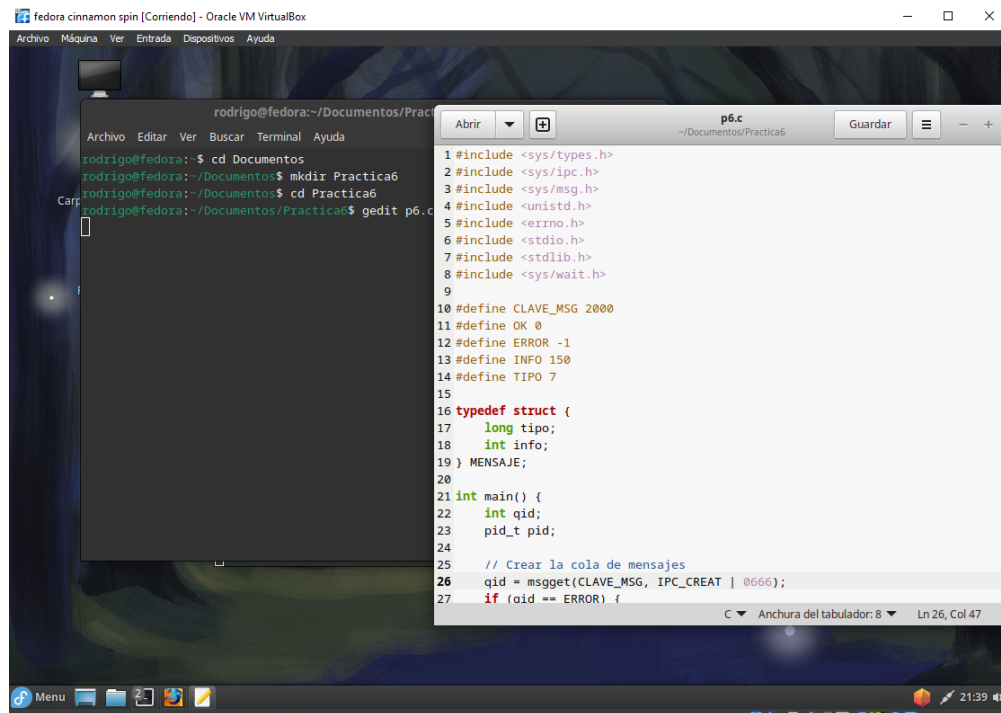
Separe el código en dos procesos (pueden ser establecidos mediante dos ejecutables o utilizando la función `fork()`) para realizar el envío de mensajes entre ambos procesos utilizando una cola de mensajes:

Código elaborado:

```
1  #include <sys/types.h>
2  #include <sys/ipc.h>
3  #include <sys/msg.h>
4  #include <unistd.h>
5  #include <errno.h>
6  #include <stdio.h>
7  #include <stdlib.h>
8  #include <sys/wait.h>
9
10 #define CLAVE_MSG 2000
11 #define OK 0
12 #define ERROR -1
13 #define INFO 150
14 #define TIPO 7
15
16 typedef struct {
17     long tipo;
18     int info;
19 } MENSAJE;
20
21 int main() {
22     int qid;
23     pid_t pid;
24
25     // Crear la cola de mensajes
26     qid = msgget(CLAVE_MSG, IPC_CREAT | 0666);
27     if (qid == ERROR) {
28         perror("msgget: ");
29         exit(errno);
30     }
31     printf("Cola de mensajes creada...\n");
32
33     // SEPARACION DE PROCESOS MEDIANTE FORK
34     pid = fork();
35
36     if (pid < 0) {
37         // Error al crear el proceso hijo
38         perror("fork: ");
39         exit(errno);
40     } else if (pid == 0) {
41         // *** CODIGO DEL PROCESO HIJO (CONSUMIDOR) ***
42         MENSAJE msg;
43         printf("Proceso hijo (Consumidor) esperando mensajes...\n");
44
45         // Leer un mensaje de la cola
46         if (msgrcv(qid, &msg, sizeof(MENSAJE) - sizeof(long), 0, 0) == ERROR) {
```

```
47     perror("msgrcv: ");
48     exit(errno);
49 }
50
51     printf("Proceso hijo recibí mensaje: tipo = %ld, info = %d\n", msg.tipo, msg.info);
52
53     // Eliminar la cola de mensajes
54     if (msgctl(qid, IPC_RMID, NULL) == ERROR) {
55         perror("msgctl: ");
56         exit(errno);
57     }
58     printf("Cola de mensajes eliminada por el proceso hijo.\n");
59 } else {
60     // *** CDIGO DEL PROCESO PADRE (PRODUCTOR) ***
61     MENSAJE msg;
62     msg.tipo = TIPO;
63     msg.info = INFO;
64
65     printf("Proceso padre (Productor) enviando mensaje...\n");
66
67     // Enviar un mensaje a la cola
68     if (msgsnd(qid, &msg, sizeof(MENSAJE) - sizeof(long), 0) == ERROR) {
69         perror("msgsnd: ");
70         exit(errno);
71     }
72     printf("Mensaje enviado por el proceso padre: tipo = %ld, info = %d\n", msg.tipo,
73           msg.info);
74
75     // Esperar a que el proceso hijo termine
76     wait(NULL);
77     printf("Proceso padre finalizado.\n");
78 }
79
80 return OK;
81 }
```

Codificamos:

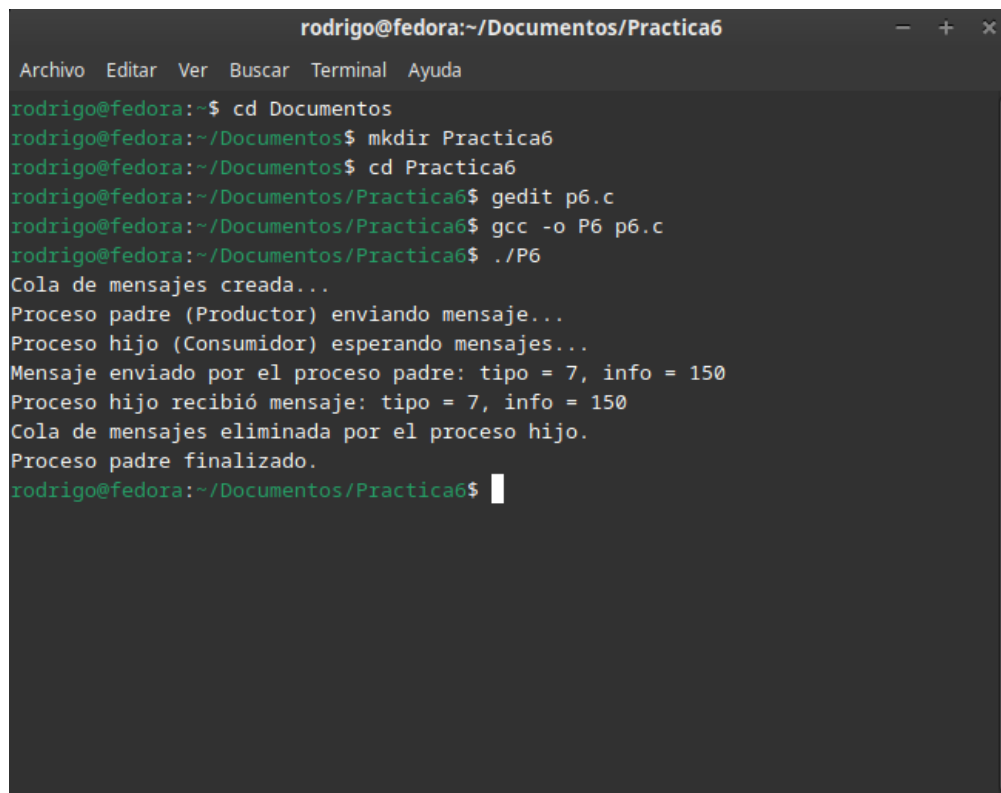


```
rodrigo@fedora:~/Documentos/Practica6
Archivo Editar Ver Buscar Terminal Ayuda
rodrigo@fedora:~$ cd Documentos
rodrigo@fedora:~/Documentos$ mkdir Practica6
rodrigo@fedora:~/Documentos$ cd Practica6
rodrigo@fedora:~/Documentos/Practica6$ gedit p6.c

p6.c
~ /Documentos/Practica6
Guardar

1 #include <sys/types.h>
2 #include <sys/ipc.h>
3 #include <sys/msg.h>
4 #include <unistd.h>
5 #include <errno.h>
6 #include <stdio.h>
7 #include <stdlib.h>
8 #include <sys/wait.h>
9
10 #define CLAVE_MSG 2000
11 #define OK 0
12 #define ERROR -1
13 #define INFO 150
14 #define TIPO 7
15
16 typedef struct {
17     long tipo;
18     int info;
19 } MENSAJE;
20
21 int main() {
22     int qid;
23     pid_t pid;
24
25     // Crear la cola de mensajes
26     qid = msgget(CLAVE_MSG, IPC_CREAT | 0666);
27     if (qid == ERROR) {
```

Ejecutamos:



```
rodrigo@fedora:~/Documentos/Practica6
Archivo Editar Ver Buscar Terminal Ayuda
rodrigo@fedora:~$ cd Documentos
rodrigo@fedora:~/Documentos$ mkdir Practica6
rodrigo@fedora:~/Documentos$ cd Practica6
rodrigo@fedora:~/Documentos/Practica6$ gedit p6.c
rodrigo@fedora:~/Documentos/Practica6$ gcc -o P6 p6.c
rodrigo@fedora:~/Documentos/Practica6$ ./P6
Cola de mensajes creada...
Proceso padre (Productor) enviando mensaje...
Proceso hijo (Consumidor) esperando mensajes...
Mensaje enviado por el proceso padre: tipo = 7, info = 150
Proceso hijo recibió mensaje: tipo = 7, info = 150
Cola de mensajes eliminada por el proceso hijo.
Proceso padre finalizado.
rodrigo@fedora:~/Documentos/Practica6$
```

3.3. Ejercicio 3

Investigue el comportamiento de la opción IPC_NOWAIT sobre el proceso de comunicación usando colas de mensajes.

La opción IPC_NOWAIT se utiliza en la API de colas de mensajes de System V para realizar operaciones no bloqueantes. Permite que un proceso que intenta enviar o recibir un mensaje no quede esperando indefinidamente si la operación no se puede completar de inmediato. Esto es útil en sistemas donde el bloqueo podría causar problemas de rendimiento o retrasos.

Comportamiento de IPC_NOWAIT:

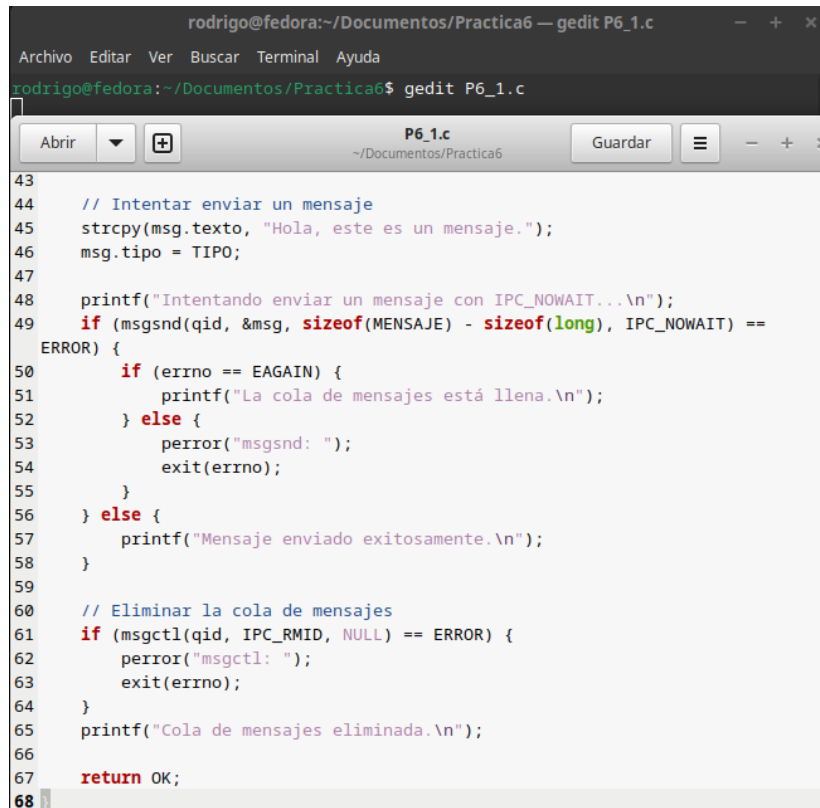
- **Recepción de mensajes (msgrcv):** si no hay mensajes disponibles que cumplan con el tipo solicitado, la función devuelve un error inmediato (-1) y establece `errno` en `ENOMSG`.
- **Envío de mensajes (msgsnd):** si la cola de mensajes está llena, la función falla inmediatamente (-1) y establece `errno` en `EAGAIN`.

Ejemplo de Implementación:

```
1  #include <sys/types.h>
2  #include <sys/ipc.h>
3  #include <sys/msg.h>
4  #include <errno.h>
5  #include <stdio.h>
6  #include <stdlib.h>
7  #include <string.h>
8
9  #define CLAVE_MSG 2000
10 #define TIPO 7
11 #define OK 0
12 #define ERROR -1
13
14 typedef struct {
15     long tipo;
16     char texto[100];
17 } MENSAJE;
18
19 int main() {
20     int qid;
21     MENSAJE msg;
22
23     // Crear o acceder a la cola de mensajes
24     qid = msgget(CLAVE_MSG, IPC_CREAT | 0666);
25     if (qid == ERROR) {
26         perror("msgget: ");
27         exit(errno);
28     }
29     printf("Cola de mensajes creada/accedida.\n");
30
31     // Intentar leer un mensaje de forma no bloqueante
32     printf("Intentando leer un mensaje con IPC_NOWAIT...\n");
33     if (msgrcv(qid, &msg, sizeof(MENSAJE) - sizeof(long), TIPO, IPC_NOWAIT) == ERROR) {
34         if (errno == ENOMSG) {
35             printf("No hay mensajes disponibles en la cola.\n");
36         } else {
37             perror("msgrcv: ");
38             exit(errno);
39         }
40     }
```

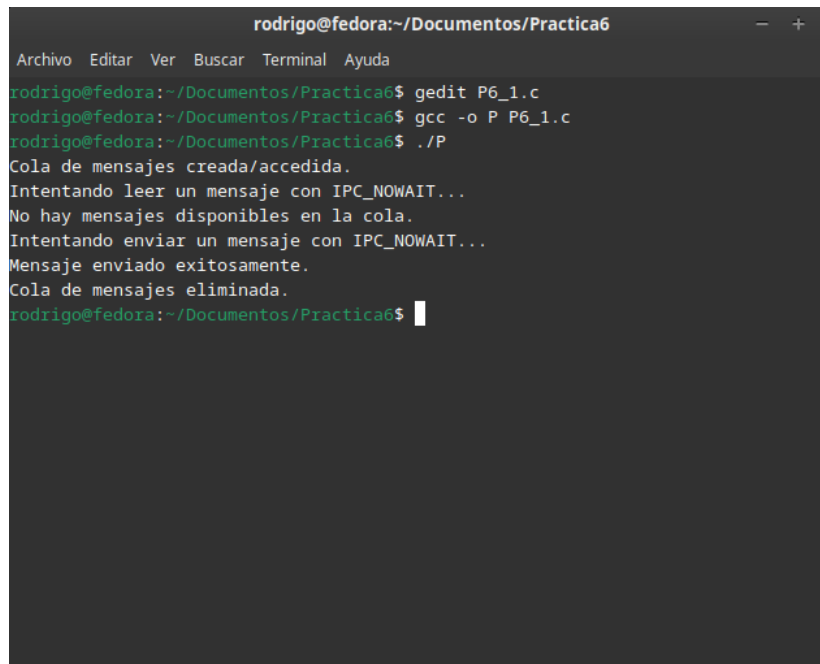
```
39     }
40 } else {
41     printf("Mensaje recibido: tipo = %ld, texto = %s\n", msg.tipo, msg.texto);
42 }
43
44 // Intentar enviar un mensaje
45 strcpy(msg.texto, "Hola, este es un mensaje.");
46 msg.tipo = TIPO;
47
48 printf("Intentando enviar un mensaje con IPC_NOWAIT...\n");
49 if (msgsnd(qid, &msg, sizeof(MENSAJE) - sizeof(long), IPC_NOWAIT) == ERROR) {
50     if (errno == EAGAIN) {
51         printf("La cola de mensajes est llena.\n");
52     } else {
53         perror("msgsnd: ");
54         exit(errno);
55     }
56 } else {
57     printf("Mensaje enviado exitosamente.\n");
58 }
59
60 // Eliminar la cola de mensajes
61 if (msgctl(qid, IPC_RMID, NULL) == ERROR) {
62     perror("msgctl: ");
63     exit(errno);
64 }
65 printf("Cola de mensajes eliminada.\n");
66
67 return OK;
68 }
```


Codificamos:



```
rodrigo@fedora:~/Documentos/Practica6 — gedit P6_1.c
Archivo Editar Ver Buscar Terminal Ayuda
rodrigo@fedora:~/Documentos/Practica6$ gedit P6_1.c
P6_1.c
~/Documentos/Practica6
Guardar
43
44 // Intentar enviar un mensaje
45 strcpy(msg.texto, "Hola, este es un mensaje.");
46 msg.tipo = TIPO;
47
48 printf("Intentando enviar un mensaje con IPC_NOWAIT...\n");
49 if (msgsnd(qid, &msg, sizeof(MENSAJE) - sizeof(long), IPC_NOWAIT) ==
ERROR) {
50     if (errno == EAGAIN) {
51         printf("La cola de mensajes está llena.\n");
52     } else {
53         perror("msgsnd: ");
54         exit(errno);
55     }
56 } else {
57     printf("Mensaje enviado exitosamente.\n");
58 }
59
60 // Eliminar la cola de mensajes
61 if (msgctl(qid, IPC_RMID, NULL) == ERROR) {
62     perror("msgctl: ");
63     exit(errno);
64 }
65 printf("Cola de mensajes eliminada.\n");
66
67 return OK;
68
```

Ejecutamos:



```
rodrigo@fedora:~/Documentos/Practica6
Archivo Editar Ver Buscar Terminal Ayuda
rodrigo@fedora:~/Documentos/Practica6$ gedit P6_1.c
rodrigo@fedora:~/Documentos/Practica6$ gcc -o P P6_1.c
rodrigo@fedora:~/Documentos/Practica6$ ./P
Cola de mensajes creada/accedida.
Intentando leer un mensaje con IPC_NOWAIT...
No hay mensajes disponibles en la cola.
Intentando enviar un mensaje con IPC_NOWAIT...
Mensaje enviado exitosamente.
Cola de mensajes eliminada.
rodrigo@fedora:~/Documentos/Practica6$
```