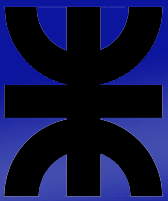


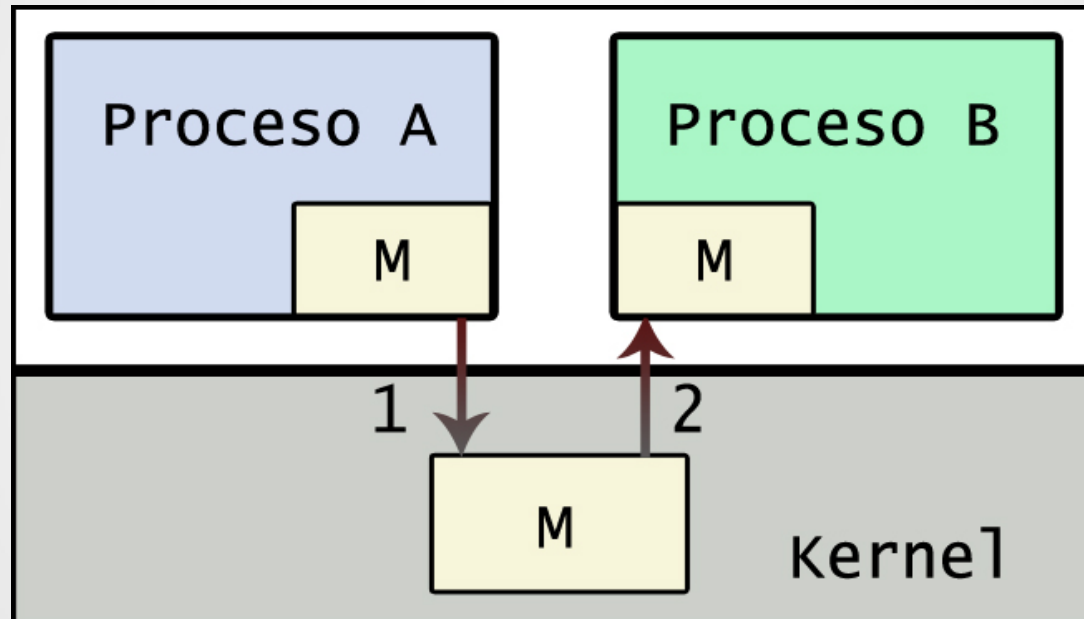
Cola de mensajes Posix



Cola de mensajes

Cola de mensajes

Permite comunicar unidades de mensajes entre procesos





Cola de mensajes

Cola de mensajes

- Permite escribir mensajes para ser leídos por diferentes procesos que conocen su identificador, por lo que podemos utilizarla en procesos no relacionados.
- Permiten a los procesos intercambio de datos en forma de mensajes.
- Los procesos introducen mensajes y se van almacenando.
- Cuando un proceso extrae un mensaje, este mensaje se borra.
- Los mensajes se ordenan por prioridad y luego para la misma prioridad por antigüedad.
- Se pueden abrir configuradas como bloqueantes o no bloqueantes.
- Las colas de mensajes tiene persistencia de kernel.
- Para compilar un proceso con cola de mensajes utilizamos la opción:
-lrt (realtime library). Ejemplo **gcc -o mq01 mq01.c -lrt**



Cola de mensajes

Apertura de una cola de mensajes

La función `mq_open()` crea una nueva cola de mensajes o abre una existente.

```
#include <mqueue.h>
```

```
mqd_t mq_open(const char *name, int oflag, ... mode_t mode, struct  
mq_attr *attr );
```

```
mq_open(nombre, banderas, permisos, attr );
```

Devuelve un descriptor de cola de mensajes en caso de éxito, o (`mqd_t`) -1 en caso de error.

Para abrir una cola de mensajes existente, solo se requieren dos argumentos (`name`, `oflag`).



Cola de mensajes

El argumento oflag es una máscara de bits que controla varios aspectos de la operación de **mq_open()**.

Flag	Description
O_CREAT	Crea una cola si esta no existe
O_RDONLY	Abre para lectura solamente
O_WRONLY	Abre para escritura solamente
O_RDWR	Abre para lectura y escritura
O_NONBLOCK	Abrir en modo de no bloqueo

El argumento modo es una máscara de bits que especifica los permisos que se colocan en la cola de un nuevo mensaje.

El argumento attr es una estructura mq_attr que especifica los atributos de la nueva cola de mensajes. Si attr es NULL, la cola se crea con atributos por defecto definidos por la aplicación.



Cola de mensajes

Cierre de una cola de mensajes:

La función `mq_close()` cierra el descriptor de cola de mensajes `mqdes`.

```
#include <mqueue.h>
```

```
int mq_close(mqd_t mqdes);
```

Devuelve 0 si tiene éxito, o -1 en caso de error.

El cierre de una cola de mensajes no la elimina.

Eliminar una cola de mensajes:

Para eliminar la cola de mensajes utilizamos `mq_unlink()`.

```
#include <mqueue.h>
```

```
int mq_unlink(const char *name);
```

Devuelve 0 si tiene éxito, o -1 en caso de error.

Marca a la cola de mensaje para ser destruida cuando todos los procesos dejen de usarla.



Cola de mensajes

Envío de mensajes:

La función `mq_send()` añade el mensaje a la cola de mensajes a la que hace referencia el descriptor `mqdes`.

```
#include <mqueue.h>
```

```
int mq_send(mqd_t mqdes, const char *msg_ptr, size_t msg_len, unsigned  
int msg_prio);
```

```
mq_send(descriptor, mensaje, tamaño del mensaje, prioridad)
```

Devuelve 0 si tiene éxito, o -1 en caso de error

El argumento `msg_len` especifica la longitud del mensaje apuntado por `msg_ptr`
Cada mensaje tiene una prioridad entero no negativo, especificado por el argumento `msg_prio` (0 la más baja prioridad)



Cola de mensajes

Recepción de mensajes:

La función `mq_receive()` elimina el mensaje más antiguo con la más alta prioridad de la cola de mensajes a que se refiere el `mqdes` y devuelve el mensaje en el buffer apuntado por `msg_ptr`.

```
#include <mqueue.h>
```

```
ssize_t mq_receive(mqd_t mqdes, char *msg_ptr, size_t  
msg_len, unsigned int *msg_prio);
```

```
mq_receive(descriptor, mensaje, tamaño máximo, prioridad);
```

Devuelve el número de bytes recibidos si tuvo éxito, o `-1` si hay error.

El argumento `msg_len` es utilizado por el proceso que recibe para especificar el número de bytes de espacio disponible. Este valor debe ser mayor o igual que el especificado en `mq_msgsize` (atributo).



Cola de mensajes

Consulta de atributos:

La función `mq_getattr()` devuelve una estructura que contiene información `mq_attr` acerca de la descripción de la cola de mensajes asociada con el descriptor `mqdes`.

```
#include <mqueue.h>
```

```
int mq_getattr(mqd_t mqdes, struct mq_attr *attr);
```

Devuelve cero si tuvo éxito y -1 en caso de error.

`mq_curmsgs` : devuelve la cantidad de mensajes que están actualmente.

`mq_maxmsg`: número máximo de mensajes.

`mq_msgsize`: tamaño máximo de mensaje (en bytes)

`mq_flags`: devuelve las banderas para la descripción de la cola de mensajes abierta (0 o `O_NONBLOCK`), asociada al descriptor `mqdes`.



Cola de mensajes

Función `mq_notify`:

Las colas de mensajes POSIX tienen la capacidad de recibir una notificación asíncrona cuando llega un mensaje en una cola previamente vacía.

La función `mq_notify()` permite a un proceso registrarse para recibir la notificación de existencia de mensajes de una cola de mensajes.

```
#include <mqueue.h>
```

```
int mq_notify(mqd_t mqdes, const struct sigevent *notification);
```

Devuelve 0 si tiene éxito, o -1 en caso de error.

El argumento `mqdes` es el descriptor de la cola de mensaje.

El argumento `notification` especifica el mecanismo por el cual el proceso ha de ser notificado.



Cola de mensajes

Estructura:

```
struct sigevent {  
    int sigev_notify;           // Método de notificación  
    int sigev_signo;           // Señal de la notificación (SIGEV_SIGNAL)  
    union sigval sigev_value;  // Valor pasado al manejador de señal  
};
```

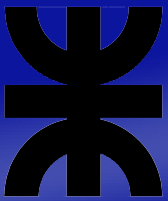
sigev_notify	Detalle
SIGEV_NONE	Registra este proceso de notificación, pero cuando llega un mensaje en la cola previamente vacía, en realidad no notifica al proceso
SIGEV_SIGNAL	Notificar al proceso mediante la generación de la señal especificada en el campo <i>sigev_signo</i> . El campo sigev_value especifica datos para acompañar a la señal.
SIGEV_THREAD	Notificar al proceso llamando a la función especificada en sigev_notify_function como si fuera la función de inicio de un nuevo hilo.



Cola de mensajes

Función `mq_notify`:

- Solo un proceso puede ser registrado para recibir una notificación de una cola de mensajes en particular.
- El proceso registrado será notificado solo cuando llega un nuevo mensaje en la cola que estaba vacía, si una cola ya contiene mensajes cuando el proceso se registra, la notificación se producirá solo después de que la cola se vacía e ingresa un mensaje nuevo.
- Después que una notificación es enviada al proceso registrado, se elimina el registro y cualquier proceso puede registrarse para ser notificado. Si el proceso desea seguir recibiendo notificaciones debe volver a registrarse después de cada notificación.
- El proceso registrado se notifica sólo si no hay otro proceso bloqueado en una llamada `mq_receive()`.



Cola de mensajes

Bibliografía

Kerrisk, Michael. *The linux programming Interface*. 2011. **Capítulo 52.**