



Escuela Politécnica Superior

Sistemas Operativos.

Práctica 3: Memoria Compartida.

Docente: Kostadin Koroutchev

Alumnos: Esaú Romo García

Daniel Cerrato Sánchez

David Teófilo Garitagoitia Romero

Carrera: Grado en Ing. Informática

18 de abril del 2021

Ejercicio 1: Creación de Memoria Compartida.

El código consiste en la creación de un segmento de memoria compartida, el cual cuenta con control de errores. Si existe un error en la creación, puede ser por dos motivos:

El primero de ellos, es en caso de que ya se haya creado el segmento de memoria compartida, en este caso se intentará abrir y si no se logra se imprimirá un mensaje que indica lo anterior.

El segundo de ellos es en caso de que haya ocurrido algún error al intentar crear el segmento de memoria ajeno a que este ya exista.

Por último, si se crea con éxito el segmento de memoria compartida, se imprimirá un mensaje de aviso.

b) Se puede colocar un “*sh_unlink*” en el `if(errno == EEXIST)` que comprueba si ya existe este segmento, de esta manera cuando este sea desocupado se elimine y quede disponible.

Ejercicio 2: Tamaño de Ficheros.

a) Se agrega la función “**fstat()**” para obtener el tamaño del segmento de memoria acompañado de un printf. Como se muestra a continuación.

```
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <unistd.h>
#include <time.h>

#define SHM_NAME "/shm_example"
#define MESSAGE "Test message"

int main(int argc, char *argv[]) {
    int fd;
    struct stat statbuf;
    size_t length = 5;
    int tam;

    if (argc != 2) {
        fprintf(stderr, "Usage: %s <FILE>\n", argv[0]);
    }

    fd = open(argv[1], O_RDWR | O_CREAT | O_EXCL, S_IRUSR | S_IWUSR);
    if (fd == -1) {
        perror("open");
        exit(EXIT_FAILURE);
    }
    dprintf(fd, "%s", MESSAGE);

    /* Get size of the file. */

    if ((fstat(fd, &statbuf)) < 0){
        perror("fstat");
        exit(EXIT_FAILURE);
    }
    printf("El tamaño en bytes del archivo es: %lu\n", statbuf.st_size);

    /* Truncate the file to size 5. */

    close(fd);
    exit(EXIT_SUCCESS);
}
```

Ilustración 1.- Uso de la función fstat() para obtener el tamaño del segmento de memoria.

b) Se agrega la función “**ftruncate()**” para truncar el segmento de memoria a 5B como se muestra a continuación.

```
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <unistd.h>
#include <time.h>

#define SHM_NAME "/shm_example"
#define MESSAGE "Test message"

int main(int argc, char *argv[]) {
    int fd;
    struct stat statbuf;
    size_t length = 5;
    int tam;

    if (argc != 2) {
        fprintf(stderr, "Usage: %s <FILE>\n", argv[0]);
    }

    fd = open(argv[1], O_RDWR | O_CREAT | O_EXCL, S_IRUSR | S_IWUSR);
    if (fd == -1) {
        perror("open");
        exit(EXIT_FAILURE);
    }
    dprintf(fd, "%s", MESSAGE);

    /* Get size of the file. */
    if ((fstat(fd, &statbuf)) < 0){
        perror("fstat");
        exit(EXIT_FAILURE);
    }

    printf("El tamaño en bytes del archivo es: %lu\n", statbuf.st_size);

    /* Truncate the file to size 5. */
    ftruncate(fd, 5);

    close(fd);
    exit(EXIT_SUCCESS);
}
```

Ilustración 2.- Utilización de la función fstat() y ftruncate.

Al ejecutar el código anterior se genera un archivo el cual contiene la palabra “**Test**” como se muestra a continuación.

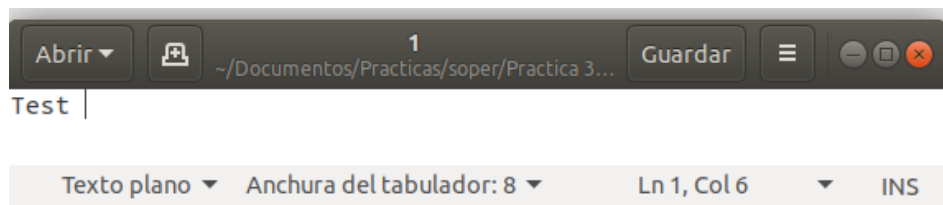
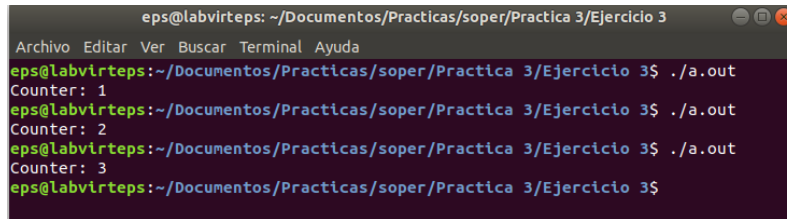


Ilustración 3.- Archivo generado.

Ejercicio 3: Mapeado de ficheros.

a) Al ejecutar varias veces el programa se aumenta el contador como se muestra en la siguiente imagen.



```
eps@labvirtpeps: ~/Documentos/Practicas/soper/Practica 3/Ejercicio 3
Archivo Editar Ver Buscar Terminal Ayuda
eps@labvirtpeps:~/Documentos/Practicas/soper/Practica 3/Ejercicio 3$ ./a.out
Counter: 1
eps@labvirtpeps:~/Documentos/Practicas/soper/Practica 3/Ejercicio 3$ ./a.out
Counter: 2
eps@labvirtpeps:~/Documentos/Practicas/soper/Practica 3/Ejercicio 3$ ./a.out
Counter: 3
eps@labvirtpeps:~/Documentos/Practicas/soper/Practica 3/Ejercicio 3$
```

Ilustración 4.- Ejecución del programa file_mmap.

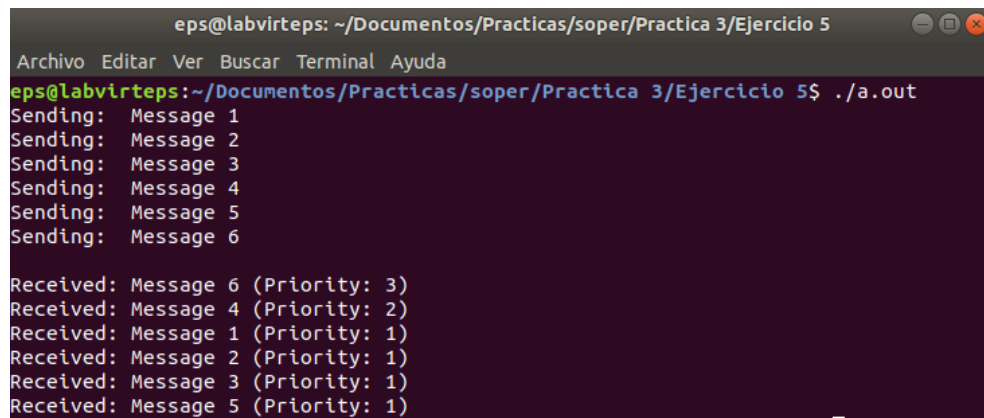
B) Se puede abrir el fichero con un editor de texto pero no se puede leer, puesto que esta toda la información en binario.

Ejercicio 4: Memoria Compartida.

- a) No, no hace falta incluirlo en el lector porque ya lo tiene el escritor. Al ejecutarlo en el escritor, hasta que no terminen de usarlo todos los procesos con acceso a la memoria compartida, no se borrará.
- b) No, ya que el escritor es el que debe de establecer el tamaño del segmento necesario de acuerdo para poder escribir en él lo que se necesite.
- c) La principal diferencia es que “**shm_open**” abre un descriptor de fichero que posteriormente puede ser mapeado, mientras que “**mmap**” solo mapea dicho segmento de memoria para que los procesos puedan acceder a él.
- d) Sí, se podría haber usado la memoria compartida, pero usando “**open**” en lugar de “**shm_open**”, el cual abre un fichero, y que el sistema operativo fuese cargando las paginas a usar en la RAM. Para acabar, se usaría “**unlink**” en lugar de “**shm_unlink**”.

Ejercicio 5: Envío y recepción de mensajes en colas.

a) El orden de envío y recepción de mensajes es el siguiente:



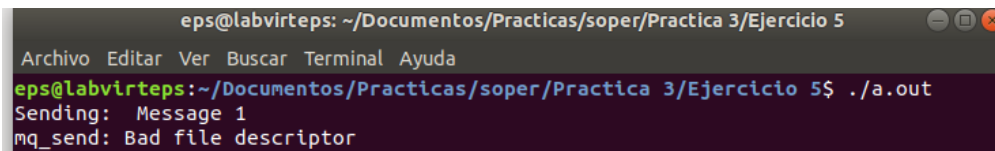
```
eps@labvirteps: ~/Documentos/Practicas/soper/Practica 3/Ejercicio 5
Archivo Editar Ver Buscar Terminal Ayuda
eps@labvirteps:~/Documentos/Practicas/soper/Practica 3/Ejercicio 5$ ./a.out
Sending: Message 1
Sending: Message 2
Sending: Message 3
Sending: Message 4
Sending: Message 5
Sending: Message 6

Received: Message 6 (Priority: 3)
Received: Message 4 (Priority: 2)
Received: Message 1 (Priority: 1)
Received: Message 2 (Priority: 1)
Received: Message 3 (Priority: 1)
Received: Message 5 (Priority: 1)
```

Ilustración 5.- envío y recepción de mensaje con cola de prioridad.

Se envían en el orden propuesto en el bucle, pero se reciben primero los de mayor prioridad y entre los de la misma prioridad, se sigue la política FIFO.

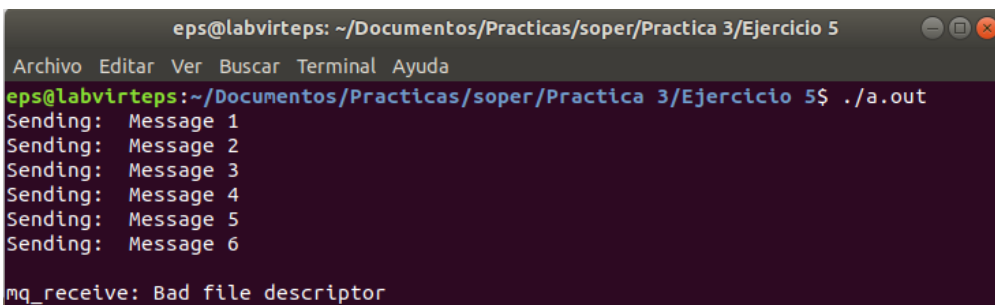
b) Al cambiar el “**O_RDWR**” por “**O_RDONLY**” provoca que la cola de mensajes sólo pueda recibir mensajes, no enviarlos, por ello se imprime el siguiente error:



```
eps@labvirteps: ~/Documentos/Practicas/soper/Practica 3/Ejercicio 5
Archivo Editar Ver Buscar Terminal Ayuda
eps@labvirteps:~/Documentos/Practicas/soper/Practica 3/Ejercicio 5$ ./a.out
Sending: Message 1
mq_send: Bad file descriptor
```

Ilustración 6.- Error al colocar O_RDONLY.

Al cambiar el “**O_RDWR**” por “**O_WRONLY**” provoca que la cola de mensajes sólo pueda enviar mensajes, no recibirlos, por ello se mandan los mensajes y se imprime el siguiente error:



```
eps@labvirteps: ~/Documentos/Practicas/soper/Practica 3/Ejercicio 5
Archivo Editar Ver Buscar Terminal Ayuda
eps@labvirteps:~/Documentos/Practicas/soper/Practica 3/Ejercicio 5$ ./a.out
Sending: Message 1
Sending: Message 2
Sending: Message 3
Sending: Message 4
Sending: Message 5
Sending: Message 6

mq_receive: Bad file descriptor
```

Ilustración 7.- Error al colocar O_WRONLY.

Ejercicio 6: Colas de mensajes.

a) Al ejecutar primero el emisor se crea una cola de mensaje y a la vez se envía el mensaje. El proceso se queda en espera hasta que se presione una tecla para terminar su ejecución. Posteriormente, al ejecutar el receptor recibe el mensaje y se imprime en pantalla, de igual forma queda en espera hasta que se presione una tecla.

b) Al ejecutar primero el receptor, este se queda en espera sin imprimir nada. Posteriormente al ejecutar el emisor, se crea la cola de mensajes y se envía la señal, provocando que en la terminal donde se ejecuta el receptor se imprima el mensaje una vez que le llegue. Ambos se quedan en espera hasta que se presione una tecla para terminar su ejecución.

c) Al crear la cola de mensajes no bloqueante usando “**O_NONBLOCK**” hace que la cola ya no se bloqué. Al ejecutar el primer inciso todo sucede de forma parecida, pero al ejecutar el segundo inciso, al encontrar la cola vacía y no ser bloqueante se devuelve un error como se muestra a continuación.

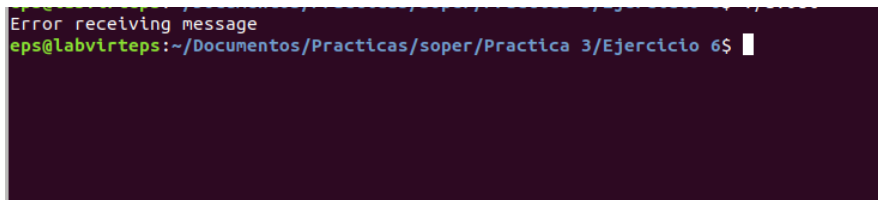
A terminal window with a dark background. The first line shows the text "Error receiving message" in red. The second line shows a green prompt "eps@labvirtips:~/Documentos/Practicas/soper/Practica 3/Ejercicio 6\$" followed by a white cursor. The rest of the terminal is empty.

Ilustración 8.- Error de mensaje al hacer la cola no bloqueante.

d) Sí, ya que se evita la condición de carrera.