Examen-Junio-2018-resuelto.pdf

Yassine_bhk

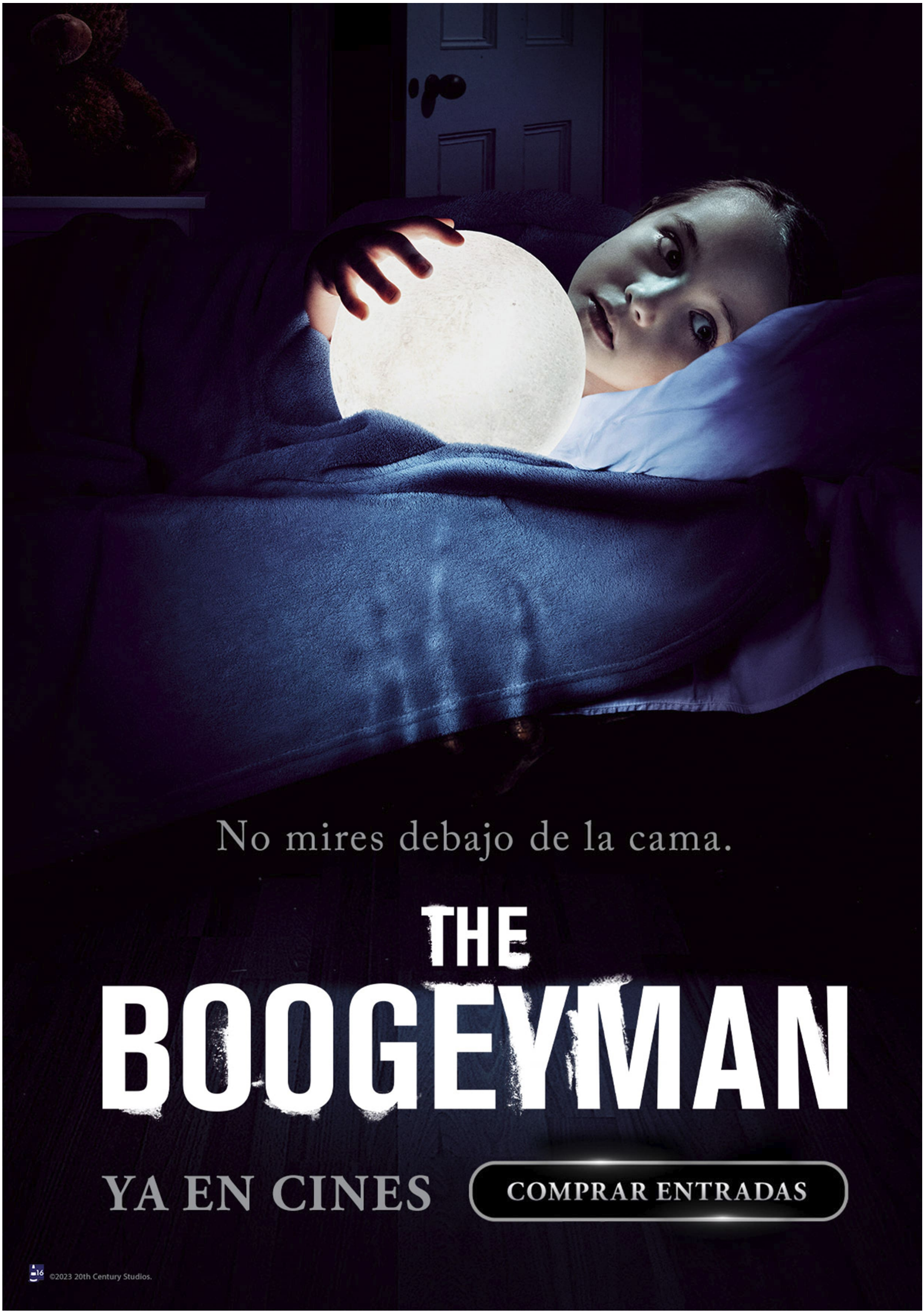
Fundamentos de Software de Comunicaciones

2º Grado en Ingeniería de las Tecnologías de Telecomunicación

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universidad de Málaga

```c
/* SERVIDOR */

#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <sys/select.h>

#define PUERTO 4950
#define MAXBUFFER 512

ssize_t read_n(int fd, void * mensaje, size_t longitud_mensaje);
ssize_t write_n(int fd, void * mensaje, size_t longitud_mensaje);


int main(int argc, char * argv[]) {
    //1.- Declaramos variables
    int sd, n_sd; //descriptores de socket
    char buffer[MAXBUFFER];
    uint8_t longitud;
    //direcciones de socket
    struct sockaddr_in serv_addr;
    struct sockaddr_in cli_addr;

    //2.- Creamos la dirección del servidor
    //2.1.- Inicializamos
    memset(&serv_addr, 0, sizeof(serv_addr));
    //2.2.- Rellenamos campos
     //2.2.1.- Familia de direcciones
     serv_addr.sin_family = AF_INET;
     //2.2.2.- Puerto
     serv_addr.sin_port = htons(PUERTO);
     //2.2.3.- Dirección IP
```

```
 serv_addr.sin_addr.s_addr = INADDR_ANY;

//4.- Creamos el socket
if ((sd = socket(PF_INET, SOCK_STREAM, 0)) < 0 ) {
    perror("socket");
    exit(1);
}

//5.- Asociamos el socket al puerto: bind()
if (bind(sd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0 ) {
    perror("bind");
    exit(1);
}

//6.- Ponemos el socket en escucha (modo pasivo): listen()
if (listen(sd, 5) < 0)  {
    perror("listen");
    exit(1);
}

//7.- Iteramos
uint16_t longBigEndian;
uint16_t dataBigEndian;
int   data;
int leidos;
while (1) {
    printf("Servidor  esperando a cliente...\n");
    socklen_t addr_len = sizeof(cli_addr);
    if ( (n_sd = accept(sd, (struct sockaddr *) &cli_addr, &addr_len )) < 0) {
        perror("accept");
        close(n_sd);
        close(sd);
        exit(1);
    } //*/

    //7.1.- Recibimos la longitud del nombre
    if (read_n(n_sd, &longitud, sizeof(longitud)) != sizeof(longitud)) {
        perror("read_n longBigEndian");
        close(n_sd);
        close(sd);
    }

    //leemos la cadena
```


No mires debajo de la cama.
THE
BOOGEYMAN
YA EN CINES
COMPRAR ENTRADAS
16
©2023 20th Century Studios.

```
if (read_n(n_sd, buffer, longitud) != longitud) {
    perror("read_n buffer");
    close(sd);
    close(n_sd);
    exit(1);
}

buffer[longitud] = '\0';
printf("fichero = %s\n", buffer);

//7.2.- abrimos el fichero
int fd;
if ((fd = open(buffer, O_WRONLY | O_CREAT | O_APPEND, 0644)) < 0) {
    perror("open");
    close(sd);
    close(n_sd);
}

printf("leyendo datos...\n");
//7.2.- Recibimos los números
leidos = read_n(n_sd, &dataBigEndian, sizeof(dataBigEndian));
while (leidos > 0) {
    if (leidos != sizeof(dataBigEndian)) {
        perror("read dataBigEndian");
        close(sd);
        close(n_sd);
        close(fd);
    }

    //convertimos a formato de host
    data = ntohs(dataBigEndian);

    printf("dato = %d\n", data);

    //escribirmos los datos en disco
    if (write_n(fd, &data, sizeof(data)) != sizeof(data)) {
        perror("write data");
        close(sd);
        close(n_sd);
        close(fd);
    }

    if (write_n(fd, "\n", 1) != 1) {
```

```
                perror("write newline");
                close(sd);
                close(n_sd);
                close(fd);
            }

            //vuelta a leer
            leidos = read_n(n_sd, &dataBigEndian, sizeof(dataBigEndian));
        }

        //comprobación de errores
        if (leidos < 0 ) {
            perror("read dataBigEndian");
            close(sd);
            close(n_sd);
            close(fd);
        }
        else { //fin de conexión
            //cierre de socket
            if (close(n_sd) < 0) {
                perror("close n_sd");
                close(sd);
                close(fd);
            }
            //cierre de fichero
            if (close(fd) < 0) {
                perror("close fd");
                close(sd);
                close(n_sd);
            }
        } // else
    } //while

    return 0;
} //main




/**
 * Funciones auxiliares
 */
```

“No necesitas besos, si no volcanes de fuego en tu boca”.
¿De quién es la frase?

a) William Shakespeare
b) Federico García Lorca
c) Nicholas Leister.

```c
ssize_t read_n(int fd, void * mensaje, size_t longitud_mensaje) {
  ssize_t a_leer = longitud_mensaje;
  ssize_t total_leido = 0;
  ssize_t leido;

  do {
    errno = 0;
    leido = read(fd, mensaje + total_leido, a_leer);
    if (leido >= 0) {
      total_leido += leido;
      a_leer -= leido;
    }
  } while((
      (leido > 0) && (total_leido < longitud_mensaje)) ||
      (errno == EINTR));

  if (total_leido > 0) {
    return total_leido;
  } else {
    /* Para permitir que devuelva un posible error en la llamada a read() */
    return leido;
  }
}

ssize_t write_n(int fd, void * mensaje, size_t longitud_mensaje) {
  ssize_t a_escribir = longitud_mensaje;
  ssize_t total_escrito = 0;
  ssize_t escrito;

  do {
    errno = 0;
    escrito = write(fd, mensaje + total_escrito, a_escribir);
    if (escrito >= 0) {
      total_escrito += escrito ;
      a_escribir -= escrito ;
    }
  } while(
        ((escrito > 0) && (total_escrito < longitud_mensaje)) ||
        (errno == EINTR));

  if (total_escrito > 0) {
```

```
    return total_escrito;
  } else {
    /* Para permitir que devuelva un posible error de la llamada a write */
    return escrito;
  }
}
```

```c
/* CLIENTE */

#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <sys/sysinfo.h>
#include <signal.h>
#include <sys/time.h>



#define PUERTO 4950
#define MAXBUFFER 512

//descriptor de socket
int sd;
struct sysinfo si;
uint16_t datoBigEndian;


ssize_t read_n(int fd, void * mensaje, size_t longitud_mensaje);
ssize_t write_n(int fd, void * mensaje, size_t longitud_mensaje);
int trimming(char * cad);


void timeout(int signo) {
    sysinfo(&si);
    short dato = si.procs;
    printf("enviando procs = %d\n", dato);

    datoBigEndian = htons(si.procs);

    if  (write_n(sd, &datoBigEndian, sizeof(datoBigEndian)) !=
sizeof(datoBigEndian)) {
        perror("write longitud");
        close(sd);
```

```
        exit(1);
    }
    signal(SIGALRM, timeout);
}

void terminar(int signo) {
    if (close(sd) < 0) {
        perror("close");
        exit(-1);
    }
    exit(0);
}


int main(int argc, char * argv[]) {
    //0.- manejadores de señal
    signal(SIGALRM, timeout);
    signal(SIGINT, terminar);


    //1.- Procesamos los argumentos
    if (argc < 3) {
        printf("Uso: %s <IP> <fichero>\n",argv[0]);
        exit(1);
    }

    //2.- Declaración de variables

    //dirección del servidor
    struct sockaddr_in serv_addr;
    //longitudes
    uint8_t longitud;

    //3.- Montamos la dirección del servidor
    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port   = htons(PUERTO);
    uint32_t ip_servidor;
    if ((ip_servidor = inet_addr(argv[1])) < 0) {
        printf("Error al convertir la IP %s\n", argv[1]);
        exit(1);
    }
    memcpy(&serv_addr.sin_addr, &ip_servidor, sizeof(ip_servidor));
```

```
//4.- Creamos el socket
if ((sd = socket(PF_INET, SOCK_STREAM, 0)) < 0 ) {
    perror("socket");
    exit(1);
}

//5.- Abrimos la conexión con el servidor
if (connect(sd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0 ) {
    perror("connect");
    exit(1);
}


//6.- Enviamos el nombre de fichero
//calculamos la longitud
longitud = strlen(argv[2]);



//la envianos (como es uint8_t no hay que pasar a big endian)
if (write_n(sd, &longitud, sizeof(longitud)) != sizeof(longitud)) {
 perror("write longitud");
 exit(1);
}

//enviamos la cadena
if  (write_n(sd, argv[2], longitud) != longitud) {
 perror("write longitud");
 exit(1);
}

struct itimerval t;
struct timeval t0;
t0.tv_sec = 2;
t0.tv_usec = 0;
struct timeval interval;
interval.tv_sec = 2;
interval.tv_usec = 0;

t.it_value = t0;
t.it_interval = interval;
```

```
    if (setitimer(ITIMER_REAL, &t, NULL) < 0 ) {
        perror("setitimer");
        exit(-1);
    }

    //7.- Enviamos todos los datos
    while(1) {
        pause();
    }


    return 0;
}

/**
 * Funciones auxiliares
 */
ssize_t read_n(int fd, void * mensaje, size_t longitud_mensaje) {
  ssize_t a_leer = longitud_mensaje;
  ssize_t total_leido = 0;
  ssize_t leido;

  do {
    errno = 0;
    leido = read(fd, mensaje + total_leido, a_leer);
    if (leido >= 0) {
      total_leido += leido;
      a_leer -= leido;
    }
  } while((
      (leido > 0) && (total_leido < longitud_mensaje)) ||
      (errno == EINTR));

  if (total_leido > 0) {
    return total_leido;
  } else {
    /* Para permitir que devuelva un posible error en la llamada a read() */
    return leido;
  }
}

ssize_t write_n(int fd, void * mensaje, size_t longitud_mensaje) {
  ssize_t a_escribir = longitud_mensaje;
```

```
  ssize_t total_escrito = 0;
  ssize_t escrito;

  do {
    errno = 0;
    escrito = write(fd, mensaje + total_escrito, a_escribir);
    if (escrito >= 0) {
      total_escrito += escrito ;
      a_escribir -= escrito ;
    }
  } while(
        ((escrito > 0) && (total_escrito < longitud_mensaje)) ||
        (errno == EINTR));

  if (total_escrito > 0) {
    return total_escrito;
  } else {
    /* Para permitir que devuelva un posible error de la llamada a write */
    return escrito;
  }
}

int trimming(char * cad) {
      int longitud = 0;
      //calculamos la longitud
     while (cad[longitud] != '\0') {
         longitud++;
     }

     if ((longitud > 0) && (cad[longitud-1] == '\n')) {
        cad[longitud -1] = '\0';
        longitud--;
    }

    return longitud;
}
```

```c
/* EJERCICIO */

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

#define EST_ESPERA_FIN1 0
#define EST_ESPERA_FIN2 1

#define EV_FIN 0
#define EV_NO_FIN 1
#define EV_TIMEOUT 2

int espera_evento (int fd) {

    int evento = -1;
    int leidos;
    int escritos;
    char buffer [512];

    fd_set conjunto;
    FD_ZERO (&conjunto);
    FD_SET (0, &conjunto);
    FD_SET (fd, &conjunto);
;
    struct timeval tiempo = {5, 500};


    int resultado = select (fd + 1, &conjunto, 0, 0, &tiempo);

        if (resultado == -1) {

            perror ("Error al hacer select");
            exit (1);
        }

        if (resultado == 0) {
```

```
    evento = EV_TIMEOUT;
}

else {

    if (FD_ISSET (0, &conjunto)) {

        memset (&tiempo, 0, sizeof (tiempo));

        leidos = read (0, buffer, 512);

            if (leidos < 0) {

                perror ("Error al leer");
                exit (1);
            }

            if (leidos == 0) {

                FD_CLR (0, &conjunto);
            }

            if (leidos > 0) {

                buffer [leidos] = '\0';

                char * q = strstr (buffer, "fin");

                    if (q == NULL) {

                        evento = EV_NO_FIN;
                    }

                    else {

                        evento = EV_FIN;
                    }
                }

            } // FD_ISSET

    if (FD_ISSET (fd, &conjunto)) {
```

```
                memset (&tiempo, 0, sizeof (tiempo));

                leidos = read (fd, buffer, 512);

                    if (leidos < 0) {

                        perror ("Error al leer");
                        exit (1);
                    }

                    if (leidos == 0) {

                        FD_CLR (fd, &conjunto);
                    }

                    if (leidos > 0) {

                        buffer [leidos] = '\0';

                        char * q = strstr (buffer, "fin");

                            if (q == NULL) {

                                evento = EV_NO_FIN;
                            }

                            else {

                                evento = EV_FIN;
                            }
                        }
                } // FD_ISSET
            } // RESULTADO > 0

    return evento;

}

int main () {

    int estado = EST_ESPERA_FIN1;
    int evento;
    int fin = 0;
```

```
int fd = open ("fsc_fifo", O_RDONLY);

    if (fd < 0) {

        perror ("Error al abrir la fifo");
        exit (1);
    }

    while (!fin) {

        evento = espera_evento (fd);

            if (evento < 0) {

                printf ("Evento no reconocido");

            }

        switch (estado)
        {
        case EST_ESPERA_FIN1 :

            switch (evento)
            {

            case EV_NO_FIN:


                break;

            case EV_FIN:

                estado = EST_ESPERA_FIN2; printf ("Ha entrado\n");

                break;

            case EV_TIMEOUT:


                break;

            }
```

```
                break;

            case EST_ESPERA_FIN2 :

                switch (evento)
                {

                case EV_NO_FIN:


                    break;

                case EV_FIN:

                    fin = 1; printf ("Ha entrado2\n");

                    break;

                case EV_TIMEOUT:

                    estado = EST_ESPERA_FIN1;


                    break;

                } // switch evento

                break;

            } //switch estado
        } // bucle while


    return 0;
}
```