Examen-Febrero-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

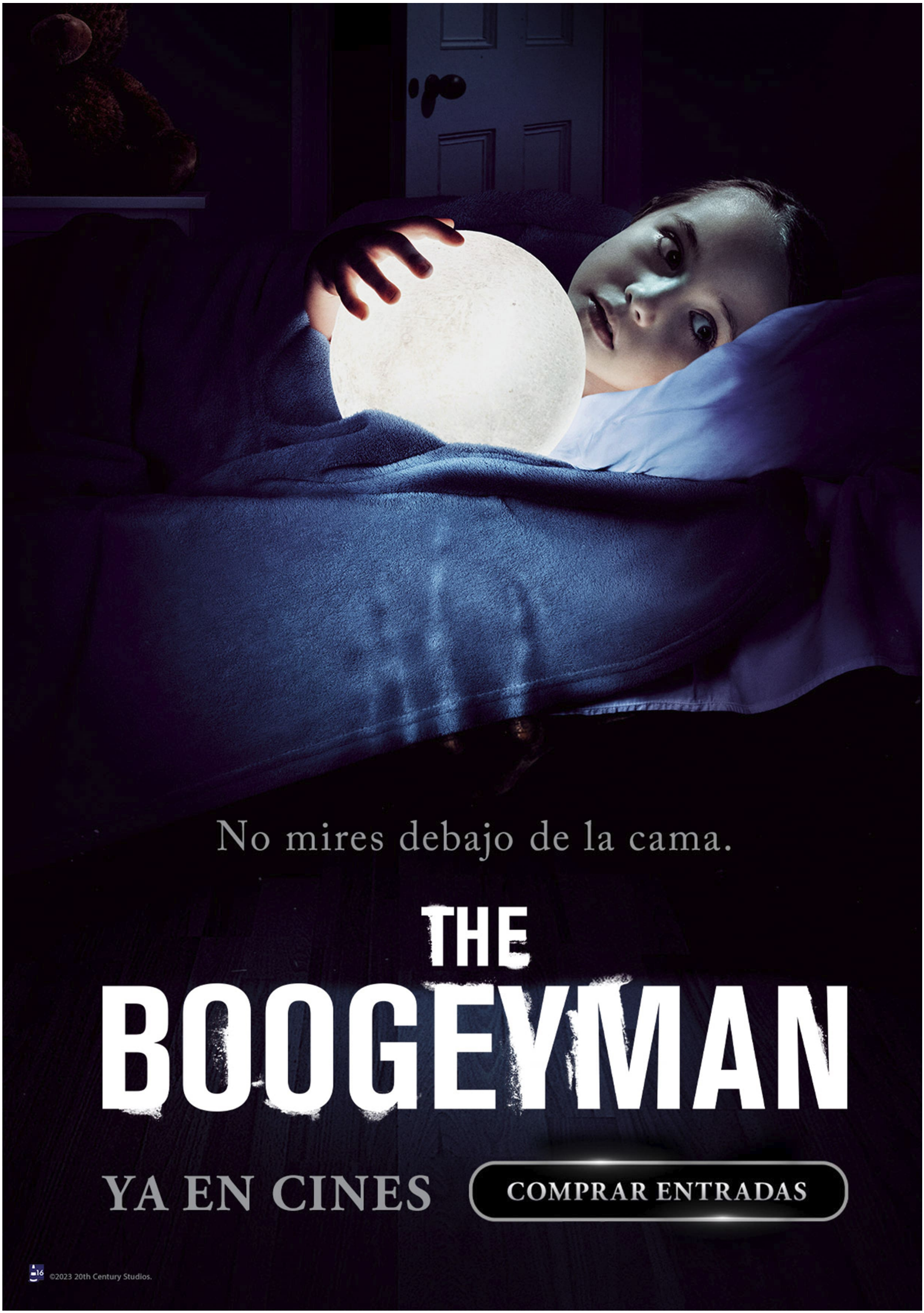
216 /* CLIENTE */

```c
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <signal.h>
#include <stdint.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <string.h>
#include <fcntl.h>

#define PUERTO 4951
#define TAM_MAX 512

ssize_t readn(int fd, char* p, size_t max){
    ssize_t leidos = 0;
    size_t a_leer = max;
    char *m = p;
    size_t r;
    do{
        errno = 0;
        r = read(fd,m + leidos,a_leer);
        if(r > 0){
            leidos += r;
            a_leer -= r;
        }
    }while(((a_leer != 0) && (r > 0)) || (errno == EINTR));
    if(r < 0){
        return -1;
    }else{
        return leidos;
    }
}
```

```
251
252 ssize_t writen(int fd, char* p, size_t max){
253     ssize_t escritos = 0;
254     size_t a_escribir = max;
255     char *m = p;
256     size_t w;
257     do{
258         errno = 0;
259         w = write(fd, m + escritos, a_escribir);
260         if(w > 0){
261             escritos += w;
262             a_escribir -= w;
263         }
264     }while(((w > 0) && (a_escribir != 0)) || (errno == EINTR));
265     if(w < 0){
266         return -1;
267     }else{
268         return escritos;
269     }
270
271 }
272
273
274 int main(int argc, char **argv){
275     if(argc < 2){
276         printf("Falta la dirección IP\n");
277         exit(1);
278     }
279
280     struct sockaddr_in vinculo;
281     int sd;
282     uint16_t leidos, longred;
283
284     sd = socket(AF_INET,SOCK_STREAM,0);
285     if(sd < 0){
286         perror("SOCKET");
287         exit(1);
288     }
289
```


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

```
290     memset(&vinculo,0,sizeof(vinculo));
291     vinculo.sin_family = PF_INET;
292     vinculo.sin_port = htons(PUERTO);
293     vinculo.sin_addr.s_addr = inet_addr(argv[1]);
294
295     if(connect(sd,(struct sockaddr*)&vinculo,sizeof(vinculo)) < 0){
296         perror("ERROR ACCEPT");
297         close(sd);
298         exit(1);
299     }
300
301     int fin = 0;
302     int leid;
303     char buffer[TAM_MAX];
304     while(!fin){
305         leidos = read(0,buffer,TAM_MAX - 4); /* Resto 4 porqe luego hay que sumarle 123 y meterle \0*/
306         if(leidos < 0){
307             perror("Error leyendo de teclado");
308             close(sd);
309             exit(1);
310         }
311         buffer[leidos-1] = '\0';
312         printf("LEIDO %s\n",buffer);
313         if(strcmp(buffer,"fin") == 0){
314             fin = 1;
315             int w = writen(sd,(char*)NULL,(size_t )NULL);
316             if(w < 0){
317                 perror("Error mandando cadena vacia");
318                 close(sd);
319                 exit(1);
320             }
321             break;
322         }
323         longred = htons(leidos);
324         int w = writen(sd,(char*)&longred,2);
```


si lees esto me debes un besito

```
325         if(w != 2){
326             perror("Error cabecera");
327             close(sd);
328             exit(1);
329         }
330         w = writen(sd,buffer,leidos);
331         if(w != leidos){
332             perror("ERROR escribiendo a servidor\n");
333             close(sd);
334             exit(1);
335         }
336         int r = readn(sd,(char*)&longred,2);
337         if(r != 2){
338             perror("Error cabecera lectura\n");
339             close(sd);
340             exit(1);
341         }
342         leidos = ntohs(longred);
343         r = readn(sd,buffer,leidos);
344         if(r != leidos){
345             perror("Error leyendo cadena codificada\n");
346             close(sd);
347             exit(1);
348         }
349         buffer[r] = '\0';
350         printf("CADENA CODIFICADA %s\n",buffer);
351 
352     }
353 
354     close(sd);
355 
356     return 0;
357 }
358 
359 
```

C: > Users > yassi > Downloads > hola.c

```c

/* SERVIDOR */

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

#define PUERTO 4951
#define TAM_MAX 512


ssize_t readn(int fd, char* p, size_t max){
    ssize_t leidos = 0;
    size_t a_leer = max;
    char *m = p;
    size_t r;
    do{
        errno = 0;
        r = read(fd,m + leidos,a_leer);
        if(r > 0){
            leidos += r;
            a_leer -= r;
        }
    }while(((a_leer != 0) && (r > 0)) || (errno == EINTR));
    if(r < 0){
        return -1;
    }else{
        return leidos;
    }
}

```

```c
ssize_t writen(int fd, char* p, size_t max){
    ssize_t escritos = 0;
    size_t a_escribir = max;
    char *m = p;
    size_t w;
    do{
        errno = 0;
        w = write(fd, m + escritos, a_escribir);
        if(w > 0){
            escritos += w;
            a_escribir -= w;
        }
    }while(((w > 0) && (a_escribir != 0)) || (errno == EINTR));
    if(w < 0){
        return -1;
    }else{
        return escritos;
    }

}

int maximo(int fd1, int fd2){
    if(fd1 > fd2){
        return fd1;
    }else{return fd2;}
}

void copiar_conjunto(fd_set *destino, const fd_set *origen, int max ){
    FD_ZERO(destino);
    for(int i = 0; i < max; i++){
        if(FD_ISSET(i,origen)){
            FD_SET(i,destino);
        }
    }
}

```

```c
int main(){

    /* Antes de aceptar clientes es necesario leer el offset por la fifo*/
    int fd_fifo;

    fd_fifo = open("fifo_admin",O_RDONLY);
    printf("FIFO abierta\n");
    if(fd_fifo < 0){
        perror("FIFO");
        exit(1);
    }

    uint8_t offset;
    char buffer[TAM_MAX];
    int leidos = read(fd_fifo,buffer,TAM_MAX-1);
    printf("LEIDOS BIEN\n");
    buffer[leidos-1] = '\0';
    if(leidos < 0){
        perror("Error leyendo offset de fifo");
        close(fd_fifo);
        exit(1);
    }
    offset = atoi(buffer);
    printf("Valor obtenido %d\n",offset);

    /* Offset leido */

    char cadena[TAM_MAX];
    uint16_t longCadena, longCadenaRed;

    /* Se procede a la declaración de sockets */
    struct sockaddr_in server, cliente;
    socklen_t longCli = sizeof(cliente);
    int sd, csd;

    sd = socket(AF_INET, SOCK_STREAM,0);
    if(sd < 0){
        perror("SOCKET");
        close(fd_fifo);
        exit(1);
```

```
118     }
119
120     memset(&server,0,sizeof(server));
121     server.sin_family = PF_INET;
122     server.sin_port = htons(PUERTO);
123     server.sin_addr.s_addr = INADDR_ANY;
124
125     int resBind = bind(sd,(struct sockaddr*)&server,sizeof(server));
126     printf("BIND HECHO\n");
127     if(resBind < 0){
128         perror("BIND");
129         close(fd_fifo);
130         exit(1);
131     }
132
133     int resListen = listen(sd,10);
134     printf("LISTEN HECHO\n");
135     if(resListen < 0){
136         perror("LISTEN");
137         close(fd_fifo);
138         exit(1);
139     }
140     int fin = 0;
141     while(1){
142         csd = accept(sd,(struct sockaddr*)&cliente,&longCli);
143         printf("ACCEPT HECHO\n");
144         if(csd < 0){
145             perror("ACCEPT");
146             close(sd);
147             close(fd_fifo);
148             exit(1);
149         }
150         fin = 0;
151
152         fd_set conjunto, conjunto_mod;
153         FD_ZERO(&conjunto);
154         FD_SET(fd_fifo,&conjunto);
155         FD_SET(csd,&conjunto);
156         int max = maximo(csd,fd_fifo);
157
```

```
158
159         while(!fin){
160             copiar_conjunto(&conjunto_mod,&conjunto,max+1);
161             int resSelect = select(max+1,&conjunto_mod,NULL,NULL,0);
162             printf("SELECT HECHO \n");
163             if(resSelect < 0){
164                 perror("Error select");
165                 close(fd_fifo);
166                 close(csd);
167                 close(sd);
168                 exit(1);
169             }
170             if(FD_ISSET(fd_fifo,&conjunto_mod)){
171                 leidos = read(fd_fifo,buffer,TAM_MAX-1);
172                 printf("LEYENDO DE FIFO\n");
173                 buffer[leidos-1] = '\0';
174                 if(leidos < 0){
175                     perror("Error leyendo offset de fifo");
176                     close(fd_fifo);
177                     exit(1);
178                 }
179                 offset = atoi(buffer);
180                 printf("Valor obtenido %d\n",offset);
181             }
182             if(FD_ISSET(csd,&conjunto_mod)){
183                 leidos = readn(csd,(char*)&longCadenaRed,2);
184                 if(leidos < 0){
185                     perror("Error leyendo longitud de cadena");
186                     close(fd_fifo);
187                     close(csd);
188                     close(sd);
189                     exit(1);
190                 }else{
191                     if(leidos == 0){
192                         fin = 1;
193                         continue;
194                     }
195                 }
196
197                 longCadena = ntohs(longCadenaRed);
```

```
198
199             leidos = readn(csd,cadena,longCadena);
200             if(leidos != longCadena){
201                 perror("Error leyendo cadena");
202                 close(fd_fifo);
203                 close(csd);
204                 close(sd);
205                 exit(1);
206             }
207
208             for(int i = 0; i < longCadena; i++){
209                 printf("%c\n",cadena[i]);
210                 cadena[i] = cadena[i] + offset;
211             }
212             cadena[longCadena]  = '\0';
213             strcat(cadena,"123");
214             uint16_t l, lred;
215             l = strlen(cadena);
216             lred = htons(l);
217             int w = writen(csd,(char*)&lred,2);
218             if(w != 2){
219                 perror("Error escribiendo cabecera cadena codificada");
220                 close(csd);
221                 close(sd);
222                 close(fd_fifo);
223                 exit(1);
224             }
225             w = writen(csd,cadena,strlen(cadena));
226             if(w != strlen(cadena)){
227                 perror("ERROR MANDANDO CADENA ENCRIPTADA");
228                 close(csd);
229                 close(sd);
230                 close(fd_fifo);
231                 exit(1);
232             }
233
234         }
235     }
236
```

```
236
237
238
239         }
240         close(csd);
241         close(sd);
242         close(fd_fifo);
243
244         return 0;
245     }
```


si lees esto me debes un besito

C: > Users > yassi > Downloads > hola.c

```c
/* EJERCICIO 1*/

#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <signal.h>


int fsc_system(char * comando) {

    int pid_hijo = fork ();

        if (pid_hijo < 0) {

            perror ("Error fork");
            exit (1);
        }

    if (pid_hijo == 0) {

        int resultado = execl ("/bin/bash", "bash", "-c", comando, 0);

            if (resultado < 0) {

                perror ("Error execl");
                exit (1);
            }
        }

    if (pid_hijo > 0) {

        wait (0);
    }
}

```


si lees esto me debes un besito

```c
int fsc_timeout(int s, char * comando) {

    int pid_hijo = fork ();

        if (pid_hijo < 0) {

            perror ("Error fork");
            exit (1);
        }

    if (pid_hijo == 0) {

        int resultado = execl ("/bin/bash", "bash", "-c", comando, 0);

            if (resultado < 0) {

                perror ("Error execl");
                exit (1);
            }
        }

    if (pid_hijo > 0) {

        sleep (s);
        kill (pid_hijo, SIGTERM);
        wait (0);
    }
}

int main (int argc, char *argv []) {

    if (argc < 3) {

        printf ("Porfavor, escriba el nombre del comando");
        exit (1);
    }

    fsc_system (argv [1]);
    fsc_timeout (atoi (argv [2]), argv [1]);


    return 0;
}

```


si lees esto me debes un besito