

# 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**



**#YoElijo  
Cerveza SIN**

**Sea cual sea  
el vehículo que  
conduces, elige  
cerveza SIN.**



**UNA GRAN CERVEZA.  
UNA GRAN RESPONSABILIDAD.**



© CONDUCCIÓN RESPONSABLE. CERVEZA SIN es una iniciativa de la Asociación de Cerveceros de España con el apoyo de la Dirección General de Tráfico.





**Invita a otros estudiantes, crea contenido y gana los premios que te alegrarán el verano**



**participa  
aquí**

216    **/\* CLIENTE \*/**

```
216  #include <stdio.h>
217  #include <unistd.h>
218  #include <errno.h>
219  #include <signal.h>
220  #include <stdint.h>
221  #include <stdlib.h>
222  #include <sys/types.h>
223  #include <sys/socket.h>
224  #include <arpa/inet.h>
225  #include <netinet/in.h>
226  #include <string.h>
227  #include <fcntl.h>
228
229  #define PUERTO 4951
230  #define TAM_MAX 512
231
232  ssize_t readn(int fd, char* p, size_t max){
233      ssize_t leidos = 0;
234      size_t a_leer = max;
235      char *m = p;
236      size_t r;
237      do{
238          errno = 0;
239          r = read(fd, m + leidos, a_leer);
240          if(r > 0){
241              leidos += r;
242              a_leer -= r;
243          }
244      }while(((a_leer != 0) && (r > 0)) || (errno == EINTR));
245      if(r < 0){
246          return -1;
247      }else{
248          return leidos;
249      }
250  }
```

```

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



©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 leido;
303     char buffer[TAM_MAX];
304     while(!fin){
305         leidos = read(0,buffer,TAM_MAX - 4); /* Resto 4 porque 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);

```

```

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

```



**Invita a otros estudiantes, crea contenido y gana los premios que te alegrarán el verano**



**participa aquí**



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

```
1
2  /* SERVIDOR */
3
4  #include <stdio.h>
5  #include <unistd.h>
6  #include <errno.h>
7  #include <signal.h>
8  #include <stdint.h>
9  #include <stdlib.h>
10 #include <sys/types.h>
11 #include <sys/time.h>
12 #include <sys/socket.h>
13 #include <arpa/inet.h>
14 #include <netinet/in.h>
15 #include <string.h>
16 #include <fcntl.h>
17
18 #define PUERTO 4951
19 #define TAM_MAX 512
20
21
22 ssize_t readn(int fd, char* p, size_t max){
23     ssize_t leidos = 0;
24     size_t a_leer = max;
25     char *m = p;
26     size_t r;
27     do{
28         errno = 0;
29         r = read(fd,m + leidos,a_leer);
30         if(r > 0){
31             leidos += r;
32             a_leer -= r;
33         }
34     }while(((a_leer != 0) && (r > 0)) || (errno == EINTR));
35     if(r < 0){
36         return -1;
37     }else{
38         return leidos;
39     }
40 }
41
```



```

42 ssize_t writen(int fd, char* p, size_t max){
43     ssize_t escritos = 0;
44     size_t a_escribir = max;
45     char *m = p;
46     size_t w;
47     do{
48         errno = 0;
49         w = write(fd, m + escritos, a_escribir);
50         if(w > 0){
51             escritos += w;
52             a_escribir -= w;
53         }
54     }while(((w > 0) && (a_escribir != 0)) || (errno == EINTR));
55     if(w < 0){
56         return -1;
57     }else{
58         return escritos;
59     }
60 }
61
62
63 int maximo(int fd1, int fd2){
64     if(fd1 > fd2){
65         return fd1;
66     }else{return fd2;}
67 }
68
69 void copiar_conjunto(fd_set *destino, const fd_set *origen, int max ){
70     FD_ZERO(destino);
71     for(int i = 0; i < max; i++){
72         if(FD_ISSET(i,origen)){
73             FD_SET(i,destino);
74         }
75     }
76 }
77

```



```

78 int main(){
79
80     /* Antes de aceptar clientes es necesario leer el offset por la fifo*/
81     int fd_fifo;
82
83     fd_fifo = open("fifo_admin",O_RDONLY);
84     printf("FIFO abierta\n");
85     if(fd_fifo < 0){
86         perror("FIFO");
87         exit(1);
88     }
89
90     uint8_t offset;
91     char buffer[TAM_MAX];
92     int leidos = read(fd_fifo,buffer,TAM_MAX-1);
93     printf("LEIDOS BIEN\n");
94     buffer[leidos-1] = '\0';
95     if(leidos < 0){
96         perror("Error leyendo offset de fifo");
97         close(fd_fifo);
98         exit(1);
99     }
100     offset = atoi(buffer);
101     printf("Valor obtenido %d\n",offset);
102
103     /* Offset leído */
104
105     char cadena[TAM_MAX];
106     uint16_t longCadena, longCadenaRed;
107
108     /* Se procede a la declaración de sockets */
109     struct sockaddr_in server, cliente;
110     socklen_t longCli = sizeof(cliente);
111     int sd, csd;
112
113     sd = socket(AF_INET, SOCK_STREAM,0);
114     if(sd < 0){
115         perror("SOCKET");
116         close(fd_fifo);
117         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

```



**Invita a otros estudiantes, crea contenido y gana los premios que te alegrarán el verano**



**participa  
aquí**

```
158
159
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);
```

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

```

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

```

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

```
1  /* EJERCICIO 1*/
2
3  #include <stdio.h>
4  #include <unistd.h>
5  #include <sys/types.h>
6  #include <unistd.h>
7  #include <stdlib.h>
8  #include <string.h>
9  #include <sys/types.h>
10 #include <sys/wait.h>
11 #include <signal.h>
12
13
14 int fsc_system(char * comando) {
15
16     int pid_hijo = fork ();
17
18     if (pid_hijo < 0) {
19
20         perror ("Error fork");
21         exit (1);
22     }
23
24     if (pid_hijo == 0) {
25
26         int resultado = execl ("/bin/bash", "bash", "-c", comando, 0);
27
28         if (resultado < 0) {
29
30             perror ("Error execl");
31             exit (1);
32         }
33     }
34
35     if (pid_hijo > 0) {
36
37         wait (0);
38     }
39 }
40
```



```

40
41 int fsc_timeout(int s, char * comando) {
42
43     int pid_hijo = fork ();
44
45     if (pid_hijo < 0) {
46
47         perror ("Error fork");
48         exit (1);
49     }
50
51     if (pid_hijo == 0) {
52
53         int resultado = execl ("/bin/bash", "bash", "-c", comando, 0);
54
55         if (resultado < 0) {
56
57             perror ("Error execl");
58             exit (1);
59         }
60     }
61
62     if (pid_hijo > 0) {
63
64         sleep (s);
65         kill (pid_hijo, SIGTERM);
66         wait (0);
67     }
68 }
69
70 int main (int argc, char *argv []) {
71
72     if (argc < 3) {
73
74         printf ("Porfavor, escriba el nombre del comando");
75         exit (1);
76     }
77
78     fsc_system (argv [1]);
79     fsc_timeout (atoi (argv [2]), argv [1]);
80
81
82     return 0;
83 }
84

```