Examen-Septiembre-2021-Resuelto.pdf

Yassine_bhk

Fundamentos de Software de Comunicaciones

2º Grado en Ingeniería de las Tecnologías de Telecomunicación

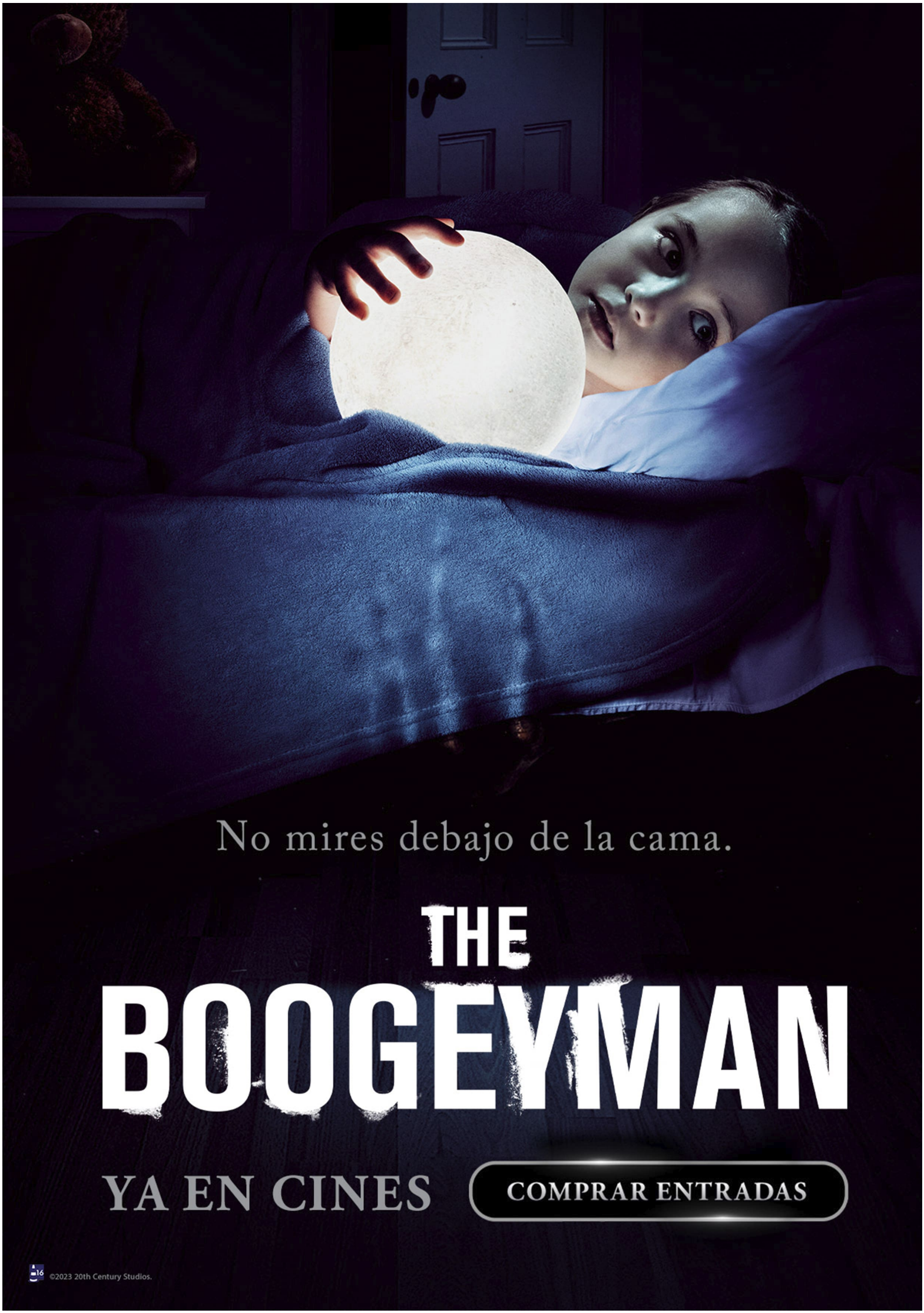
Escuela Técnica Superior de Ingeniería de Telecomunicación
Universidad de Málaga

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

```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/wait.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <sys/select.h>
#include <sys/time.h>
#include <sys/sysinfo.h>
#include <signal.h>

#define PUERTO 5050


ssize_t readn (int fd, void * buffer, size_t n) {

    char * p = (char *) buffer;
    int leidos;
    int total_leidos = 0;
    int intentando_leer = n;

    do {

        errno = 0;
        leidos = read (fd, p + total_leidos, intentando_leer);

            if (leidos > 0) {

                total_leidos += leidos;
                intentando_leer -= leidos;
```

```
41              }
42
43      } while (leidos > 0 && total_leidos < n || errno == EINTR);
44
45      if (leidos < 0) return leidos;
46      else return total_leidos;
47  }
48
49  ssize_t writen (int fd, void * buffer, size_t n) {
50
51      char * p = (char *) buffer;
52      int leidos;
53      int total_leidos = 0;
54      int intentando_leer = n;
55
56      do {
57
58          errno = 0;
59          leidos = write (fd, p + total_leidos, intentando_leer);
60
61              if (leidos > 0) {
62
63                  total_leidos += leidos;
64                  intentando_leer -= leidos;
65              }
66
67      } while ((leidos > 0 && total_leidos < n) && errno == EINTR);
68
69      if (leidos < 0) return leidos;
70      else return total_leidos;
71  }
72
73
74  int main (int argc, char * argv []) {
75
76      if (argc < 2) {
77
78          printf ("Escriba algun argumento porfavor");
79          exit (1);
80      }
```


si lees esto me debes un besito


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

```
81
82      int sd = socket (AF_INET, SOCK_STREAM, 0);
83
84          if (sd < 0) {
85
86              perror ("Socket");
87              exit (1);
88          }
89
90      struct sockaddr_in vinculo;
91      vinculo.sin_addr.s_addr = inet_addr("127.0.0.1");
92      vinculo.sin_family = PF_INET;
93      vinculo.sin_port = htons (PUERTO);
94
95      if (connect (sd, (struct sockaddr *) &vinculo, sizeof (vinculo)) < 0) {
96
97          perror ("Connect:");
98          exit (1);
99      }
100
101     if (argc == 2) {
102
103         printf ("Ha introducido usted un angulo de Giro.\n");
104         int16_t datos = atoi (argv [1]);
105         uint16_t tipo = 50;
106
107         uint16_t tipo_n = htons (tipo);
108         int16_t datos_n = htons (datos);
109
110         printf ("Enviando mensaje de tipo Giro...\n");
111
112         int escritos = writen (sd, &tipo_n, 2);
113         escritos = writen (sd, &datos_n, 2);
114
115             if (escritos < 0) {
116
117                 perror ("Escribiendo datos");
118                 close (sd);
119                 exit (1);
```


si lees esto me debes un besito

```
120             }
121         } // argc == 2
122
123     if (argc == 3) {
124
125         printf ("Ha introducido usted un Mensaje de tipo Movimiento.\n");
126         int16_t datos1 = atoi (argv [1]);
127         int16_t datos2 = atoi (argv [2]);
128         uint16_t tipo = 70;
129
130         uint16_t tipo_n = htons (tipo);
131         int16_t datos1_n = htons (datos1);
132         int16_t datos2_n = htons (datos2);
133
134         printf ("Enviando mensaje de tipo Movimiento...\n");
135         int escritos = writen (sd, &tipo_n, 2);
136         escritos = writen (sd, &datos1_n, 2);
137         escritos = writen (sd, &datos2_n, 2);
138
139             if (escritos < 0) {
140
141                 perror ("Escribiendo datos");
142                 close (sd);
143                 exit (1);
144             }
145         } // argc == 3
146
147     if (argc == 4) {
148
149         printf ("Ha introducido usted un mensaje de tipo Contenedor.\n");
150
151         int16_t datos1 = atoi (argv [1]);
152         int16_t datos2 = atoi (argv [2]);
153         int16_t datos3 = atoi (argv [3]);
154
155         uint16_t tipo = 1024;
156         uint16_t longitud = 2;
157
158         uint16_t tipo_n = htons (tipo);
159         uint16_t longitud_n = htons (longitud);
```

```
160         int16_t datos1_n = htons (datos1);
161         int16_t datos2_n = htons (datos2);
162         int16_t datos3_n = htons (datos3);
163
164         //Tipo y longitud del mensaje
165
166         int escritos = writen (sd, &tipo_n, 2);
167         escritos = writen (sd, &longitud_n, 2);
168
169         //Mensaje de tipo Giro
170
171         uint16_t tipo_giro = 50;
172         uint16_t tipo_giro_n = htons (tipo_giro);
173         escritos = writen (sd, &tipo_giro_n, 2);
174         escritos = writen (sd, &datos1_n, 2);
175
176         //Mensaje de tipo movimiento
177
178         uint16_t tipo_mvto = 70;
179         uint16_t tipo_mvto_n = htons (tipo_mvto);
180         escritos = writen (sd, &tipo_mvto_n, 2);
181         escritos = writen (sd, &datos2_n, 2);
182         escritos = writen (sd, &datos3_n, 2);
183
184             if (escritos < 0) {
185
186                 perror ("Escribiendo datos");
187                 close (sd);
188                 exit (1);
189             }
190         } // argc == 3
191
192
193     close (sd);
194
195     return 0;
196 }
197
```

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

```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>
#include <sys/time.h>
#include <sys/sysinfo.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/wait.h>


#define PUERTO 5050


ssize_t readn (int fd, void * buffer, size_t n) {

    char * p = (char *) buffer;
    int leidos;
    int total_leidos = 0;
    int intentando_leer = n;

    do {

        errno = 0;
        leidos = read (fd, p + total_leidos, intentando_leer);

            if (leidos > 0) {
```


si lees esto me debes un besito

```
41
42                  total_leidos += leidos;
43                  intentando_leer -= leidos;
44              }
45
46      } while (leidos > 0 && total_leidos < n || errno == EINTR);
47
48      if (leidos < 0) return leidos;
49      else return total_leidos;
50  }
51
52  ssize_t writen (int fd, void * buffer, size_t n) {
53
54      char * p = (char *) buffer;
55      int leidos;
56      int total_leidos = 0;
57      int intentando_leer = n;
58
59      do {
60
61          errno = 0;
62          leidos = write (fd, p + total_leidos, intentando_leer);
63
64              if (leidos > 0) {
65
66                  total_leidos += leidos;
67                  intentando_leer -= leidos;
68              }
69
70      } while ((leidos > 0 && total_leidos < n) && errno == EINTR);
71
72      if (leidos < 0) return leidos;
73      else return total_leidos;
74  }
75
76  int num_clientes = 0;
77  int num_fallos = 0;
78
79  void manejadora (int s) {
80
```

```
81      wait (0);
82      signal (s, manejadora);
83  }
84
85  void manejadora_usr1 (int s) {
86
87      num_fallos++;
88      signal (s, manejadora_usr1);
89  }
90
91  int main () {
92
93      signal (SIGCHLD, manejadora);
94      signal (SIGUSR1, manejadora_usr1);
95
96      int sd = socket (AF_INET, SOCK_STREAM, 0);
97
98          if (sd < 0) {
99
100             perror ("Socket");
101             exit (1);
102         }
103
104     struct sockaddr_in vinculo;
105     vinculo.sin_addr.s_addr = INADDR_ANY;
106     vinculo.sin_family = PF_INET;
107     vinculo.sin_port = htons (PUERTO);
108
109     if (bind (sd, (struct sockaddr *) &vinculo, sizeof (vinculo)) < 0) {
110
111         perror ("Bind:");
112         exit (1);
113     }
114
115     if (listen (sd, 5)) {
116
117         perror ("Listen");
118         exit (1);
119     }
```

```
120
121     struct sockaddr_in info_cliente;
122     socklen_t tam = sizeof (info_cliente);
123
124     do {
125
126         int csd = accept (sd, (struct sockaddr *) &info_cliente, &tam);
127         printf ("Nuevo cliente, cuyo descriptor de fichero es: %d\n", csd);
128
129             if (csd == -1) {
130
131                 /*if (errno == EINTR) {
132
133                     errno = 0;
134                 }
135
136                 else {*/
137
138                     perror ("Accept");
139                     close (sd);
140                     exit (1);
141                 //}
142             }
143
144         num_clientes ++;
145
146         int pid_hijo = fork ();
147
148             if (pid_hijo == 0) { //Zona hijo
149
150             uint16_t tipo, tipo_n;
151             int leidos, escritos;
152             char buffer [512];
153
154
155             leidos = readn (csd, &tipo_n, 2);
156
157                 if (leidos < 0) {
158
159                     perror ("Leyendo el tipo del mensaje");
```

```
160                 close (csd);
161                 close (sd);
162                 exit (1);
163             }
164
165         tipo = ntohs (tipo_n);
166
167         if (tipo == 50) {
168
169             printf ("Mensaje recibido de tipo Giro.\n");
170
171             int16_t datos, datos_n;
172
173             leidos = readn (csd, &datos_n, 2);
174
175                 if (leidos < 0) {
176
177                 perror ("Leyendo datos cuando el tipo es 50");
178                 close (csd);
179                 close (sd);
180                 exit (1);
181             }
182
183             datos = ntohs (datos_n);
184
185             printf("Tipo de mensaje: %d\n", tipo);
186             printf("Datos: %d\n", datos);
187         }
188
189         else if (tipo == 70) {
190
191             printf ("Mensaje recibido de tipo Movimiento.\n");
192
193             int16_t datos1, datos2, datos1_n, datos2_n;
194
195             leidos = readn (csd, &datos1_n, 2);
196             leidos = readn (csd, &datos2_n, 2);
197
198                 if (leidos < 0) {
```

```
200                 perror ("Leyendo cuando el tipo es 70");
201                 close (csd);
202                 close (sd);
203                 exit (1);
204             }
205 
206             datos1 = ntohs (datos1_n);
207             datos2 = ntohs (datos2_n);
208 
209             printf("Tipo de mensaje: %d", tipo);
210             printf("Datos: %d %d\n", datos1, datos2);
211 
212         }
213 
214         else if (tipo == 1024) {
215 
216             printf ("Mensaje recibido de tipo Contenedor.\n");
217 
218             uint16_t tipo_giro, tipo_giro_n, tipo_mvto, tipo_mvto_n;
219             int16_t datos1, datos2, datos3, datos1_n, datos2_n, datos3_n;
220             uint16_t longitud_n, longitud;
221 
222             leidos = readn (csd, &longitud_n, 2);
223             longitud = ntohs (longitud_n);
224 
225             //Leo el mensaje de tipo giro
226 
227             leidos = readn (csd, &tipo_giro_n, 2);
228             leidos = readn (csd, &datos1_n, 2);
229 
230             //Leo el mensaje de tipo movimiento
231 
232             leidos = readn (csd, &tipo_mvto_n, 2);
233             leidos = readn (csd, &datos2_n, 2);
234             leidos = readn (csd, &datos3_n, 2);
235 
236             tipo_giro = ntohs (tipo_giro_n);
237             datos1 = ntohs (datos1_n);
238 
```


si lees esto me debes un besito

```
239                 tipo_mvto = ntohs (tipo_mvto_n);
240                 datos2 = ntohs (datos2_n);
241                 datos3 = ntohs (datos3_n);
242
243                     if (leidos < 0) {
244
245                     perror ("Leyendo cuando el tipo es 1024");
246                     close (csd);
247                     close (sd);
248                     exit (1);
249                 }
250
251                 printf("Tipo de mensaje: %d\n", tipo);
252                 printf("Longitud de mensaje: %d\n", longitud);
253                 printf("Tipo de mensaje  1: %d\n", tipo_giro);
254                 printf("Datos mensaje 1: %d\n", datos1);
255                 printf("Tipo de mensaje  2: %d\n", tipo_mvto);
256                 printf("Datos mensaje 2: %d %d\n", datos2, datos3);
257
258             }
259
260                close (csd);
261                exit (0);
262             }
263
264
265     } while (num_clientes < 3);
266
267     close (sd);
268     printf ("Numero de fallos: %s\n", num_fallos);
269
270     return 0;
271 }
272
273
```

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

```c
/* EJERCICIO 1*/

#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>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

#define EST_INICIAL 0
#define EST_WAITING 1
#define EST_CONNECTED 2

#define EV_CONNECT_REQUEST 0
#define EV_TIMEOUT 1
#define EV_CONNECT_CONFIRM 2
#define EV_RESET 3

ssize_t readn (int fd, void * buffer, size_t n) {

    int leidos;
    int intentando_leer = n;
    int total_leidos = 0;
    char * p = (char *) buffer;

    do {

        errno = 0;
        leidos = read (fd, p + total_leidos, intentando_leer);
```

```
41              if (leidos > 0) {
42
43                  total_leidos += leidos;
44                  intentando_leer -= leidos;
45              }
46          } while (leidos > 0 && total_leidos < n || errno == EINTR);
47
48          if (leidos < 0) {
49
50              return leidos;
51          }
52
53          else return total_leidos;
54  }
55
56  ssize_t writen (int fd, void * buffer, size_t n) {
57
58      int leidos;
59      int intentando_leer = n;
60      int total_leidos = 0;
61      char * p = (char *) buffer;
62
63      do {
64
65          errno = 0;
66          leidos = write (fd, p + total_leidos, intentando_leer);
67
68              if (leidos > 0) {
69
70                  total_leidos += leidos;
71                  intentando_leer -= leidos;
72              }
73          } while (leidos > 0 && total_leidos < n || errno == EINTR);
74
75          if (leidos < 0) {
76
77              return leidos;
78          }
79
80          else return total_leidos;
```

```
81  }
82
83  int pipe_fd [2];
84  int fin = 0;
85  int fifo;
86
87  void manejadora_alarm (int s) {
88
89      int evento = EV_TIMEOUT;
90      if (writen (pipe_fd [1], &evento, 4) < 0) {
91
92          perror ("Error al escribir");
93          fin = 1;
94          exit (1);
95      }
96
97      signal (s, manejadora_alarm);
98  }
99
100 void manejadora_usr1 (int s) {
101
102     int evento = EV_CONNECT_REQUEST;
103     if (writen (pipe_fd [1], &evento, 4) < 0) {
104
105         perror ("Error al escribir");
106         fin = 1;
107         exit (1);
108     }
109
110     signal (s, manejadora_usr1);
111
112 }
113
114 void manejadora_usr2 (int s) {
115
116     int evento = EV_CONNECT_CONFIRM;
117     if (writen (pipe_fd [1], &evento, 4) < 0) {
118
119         perror ("Error al escribir");
120         fin = 1;
```

```
121         exit (1);
122     }
123     signal (s, manejadora_usr2);
124 
125 }
126 
127 int maximo (int a, int b) {
128 
129     if (a > b) {
130 
131         return a;
132     }
133 
134     else {
135 
136         return b;
137     }
138 }
139 
140 int espera_evento () {
141 
142     fd_set conjunto;
143     FD_ZERO (&conjunto);
144     FD_SET (pipe_fd [0], &conjunto);
145     FD_SET (fifo, &conjunto);
146 
147     int max = maximo (fifo, pipe_fd [0]);
148 
149     if (select (max + 1, &conjunto, 0, 0, 0) < 0) {
150 
151         if (errno == EINTR) {
152 
153             errno = 0;
154         }
155 
156         else {
157 
158             perror ("Select");
159             exit (1);
```

```
160         }
161     }
162
163     if (FD_ISSET (pipe_fd [0], &conjunto)) {
164
165             int leidos;
166             int evento;
167
168             leidos = readn (pipe_fd [0], &evento, 4);
169
170                 if (leidos < 0) {
171
172                     if (errno == EINTR) {
173
174                         errno = 0;
175                     }
176
177                     else {
178
179                         perror ("Leyendo de la pipe");
180                         exit (1);
181                     }
182                 }
183
184             return evento;
185
186         }
187
188     if (FD_ISSET (fifo, &conjunto)) {
189
190         int leidos;
191         char buffer [512];
192
193         do {
194
195             leidos = read (fifo, buffer, 512);
196
197                 if (leidos < 0) {
198
```

```
199                     if (errno == EINTR) {
200
201                         errno = 0;
202                     }
203
204                     else {
205
206                         perror ("Leyendo de la fifo");
207                         exit (1);
208
209                     }
210                 }
211
212                 if (leidos == 0) {
213
214                     printf ("No hay nda que leer, terminando...");
215                     exit (1);
216                 }
217
218                 return EV_RESET;
219
220             } while (leidos > 0);
221
222         }
223
224 }
225
226 int main () {
227
228     printf ("El pid de este proceso es: %d\n", getpid ());
229
230     signal (SIGALRM, manejadora_alarm);
231     signal (SIGUSR1, manejadora_usr1);
232     signal (SIGUSR2, manejadora_usr2);
233
234     fifo = open ("fcs_fifo", O_RDONLY);
235
236         if (fifo < 0) {
237
238             perror ("open");
```

```
239             exit (1);
240         }
241
242     int r = pipe (pipe_fd);
243
244         if (r < 0) {
245
246             perror ("Pipe");
247             exit (1);
248         }
249
250     int estado = EST_INICIAL;
251     printf ("La máquina arranca en estado INICIAL...\n");
252
253     while (!fin) {
254
255         int evento = espera_evento ();
256
257             if (evento == -1) {
258
259                 printf ("Error en espera evento.\n");
260                 exit (1);
261             }
262
263         switch (estado)
264         {
265         case EST_INICIAL:
266
267             if (evento == EV_CONNECT_REQUEST) {
268
269                 estado = EST_WAITING;
270                 printf("Pasando al estado waiting...\n");
271                 alarm (2);
272             }
273
274             break;
275
276         case EST_WAITING:
277
```

```
278             if (evento == EV_CONNECT_CONFIRM) {
279 
280                 alarm (0);
281                 estado = EST_CONNECTED;
282                 printf("Pasando al estado conectado...\n");
283 
284             }
285 
286             else if (evento == EV_TIMEOUT) {
287 
288                 estado = EST_INICIAL;
289                 printf ("Timeout: Volviendo al estado inicial...\n");
290             }
291 
292 
293             break;
294 
295         case EST_CONNECTED:
296 
297             if (evento == EV_RESET) {
298 
299                 estado = EST_INICIAL;
300                 printf("Pasando al estado inicial...\n");
301 
302             }
303 
304             break;
305 
306         }
307     }
308 
309     return 0;
310 
311 }
312 
```