~\Desktop\infinito.c

```c
//INFINITO

//Author
// Carolina Minyu De Jesús and Lidia Villarreal

#include <signal.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
#include <string.h>
struct f{
    int forkH1, forkH2, forkN2, forkH3, forkN3, forkH4,forkP,forkDestino;
}f;

int vueltas;

void terminarHijos (int signum){

    int statusH2,statusH1,statusH3,statusH4;

    if (kill(f.forkH1, SIGTERM)==-1) perror("Error al matar H1");
    if (kill(f.forkH2, SIGTERM)==-1) perror("Error al matar H2");
    if (kill(f.forkH3, SIGTERM)==-1) perror("Error al matar H3");
    if (kill(f.forkH4, SIGTERM)==-1) perror("Error al matar H4");

    if(waitpid(f.forkH1,&statusH1,0)==-1)
        perror("Error.\n");
    else {
        printf("Proceso H1 ha terminado.\n");
    }

    if(waitpid(f.forkH2,&statusH2,0)==-1)
        perror("Error.\n");
    else {
        printf("Proceso H2 ha terminado.\n");
    }

    if(waitpid(f.forkH3,&statusH3,0)==-1)
        perror("Error.\n");
    else {
        printf("Proceso H3 ha terminado.\n");
    }

    if(waitpid(f.forkH4,&statusH4,0)==-1)
        perror("Error.\n");
    else {
        printf("Proceso H4 ha terminado.\n");
    }
    printf ("La senial ha dado %d vueltas", vueltas);
    exit(0);
}

void terminarN2 (int signum){
    int statusN2;
    if (kill(f.forkN2, SIGTERM)==-1) perror("Error al matar N2");
    if(waitpid(f.forkN2,&statusN2,0)==-1)
        perror("Error.\n");
    else {
        printf("Proceso N2 ha terminado.\n");
    }
```

```
62      exit(0);
63  }
64
65  void terminarN3 (int signum){
66      int statusN3;
67      if (kill(f.forkN3, SIGTERM)==-1) perror("Error al matar N3");
68
69      if(waitpid(f.forkN3,&statusN3,0)==-1)
70          perror("Error.\n");
71      else {
72          printf("Proceso N3 ha terminado.\n");
73      }
74      exit(0);
75  }
76
77  void manejadoraSIGUSR1(int s) {
78      if (kill(f.forkDestino, SIGUSR1)==-1)
79          perror("Error\n");
80  }
81
82
83  void manejadoraPadre(int s) {
84      //si recibe SIGUSR1 el padre envia SIGUSR2
85      if(s==SIGUSR1){
86          if (kill(f.forkH2, SIGUSR2)==-1)
87              perror("Error\n");
88      }
89      //si recibe SIGUSR2 el padre envia SIGUSR1
90      if(s==SIGUSR2){
91          if (kill(f.forkH3, SIGUSR1)==-1)
92              perror("Error\n");
93      //incrementar las vueltas
94      vueltas++;
95      }
96  }
97
98  void manejadoraSIGUSR2(int s) {
99      if (kill(f.forkDestino, SIGUSR2)==-1)
100         perror("Error \n");
101 }
102
103
104 int main(int argc, char *argv[])
105 {
106
107     struct sigaction sa;
108     sigset_t conj;
109     int flag=0;
110
111     f.forkP=getpid();
112
113
114     if(-1 == sigfillset(&conj) ) return -1;
115     if(-1 == sigprocmask(SIG_BLOCK, &conj, NULL) ) return -1;
116
117     switch (f.forkH1 = fork())
118     {
119     case -1: //ERROR
120         perror("prueba\n");
121         return -1;
122     case 0: //HIJO 1
123         printf("Soy H1, PID %d, padre %d\n", getpid(), getppid());
124         f.forkDestino=f.forkP;
125         sigdelset (&conj,SIGUSR2); //Quitamos SIGUSR2 de conj
```

```
126         sigdelset (&conj,SIGTERM);
127         sa.sa_handler = manejadoraSIGUSR2;
128         sigemptyset(&sa.sa_mask);
129         sa.sa_flags=0;
130         if(sigaction(SIGUSR2, &sa, NULL)==-1) return -1;
131 
132         while (flag == 0)
133             sigsuspend (&conj);
134         break;
135 
136     default://Padre
137         printf ("Soy el padre PID %d\n", f.forkP);
138         switch (f.forkH4 = fork())
139         {
140         case -1:
141             perror("prueba\n");
142             return -1;
143         case 0: //H4
144             printf("Soy H4, PID %d, padre %d \n", getpid(), getppid ());
145             f.forkDestino=f.forkP;
146             sigdelset (&conj,SIGUSR1); //Quitamos SIGUSR1 de conj
147             sigdelset (&conj,SIGTERM);
148             f.forkDestino=f.forkP;
149             sa.sa_handler = manejadoraSIGUSR1;
150             sigemptyset(&sa.sa_mask);
151             sa.sa_flags=0;
152             if(sigaction(SIGUSR1, &sa, NULL)==-1) return -1;
153 
154             while (flag == 0)
155                 sigsuspend (&conj);
156 
157             break;
158 
159         default: //Padre
160             switch (f.forkH2 = fork())
161             {
162             case -1:
163                 perror("prueba\n");
164                 return -1;
165             case 0: //H2
166                 sigdelset (&conj,SIGUSR2); //Quitamos SIGUSR2 de conj
167                 sigdelset (&conj,SIGTERM);
168                 sa.sa_handler = manejadoraSIGUSR2;
169                 sigemptyset(&sa.sa_mask);
170                 sa.sa_flags=0;
171                 if(sigaction(SIGUSR2, &sa, NULL)==-1) return -1;
172 
173                 switch (f.forkN2 = fork())
174                 {
175                 case -1:
176                     perror("prueba\n");
177                     break;
178                 case 0:
179                     printf("Soy N2, PID %d, padre %d \n", getpid(), getppid ());
180                     f.forkDestino=f.forkH1;
181                     while (flag == 0)
182                         sigsuspend (&conj);
183                     break;
184                 } //fin switch N2
185                 printf("Soy H2, PID %d, padre %d \n", getpid(), getppid ());
186                 f.forkDestino=f.forkN2;
187                 sa.sa_handler = terminarN2;
188                 if(sigaction(SIGTERM, &sa, NULL)==-1) return -1;
189                 while (flag == 0)
```

```
190                 sigsuspend (&conj);
191             break;
192 
193         default:
194             switch (f.forkH3 = fork())  //crear H3
195             {
196             case -1:
197                 perror("prueba\n");
198                 return -1;
199             case 0: //H3
200                 sigdelset (&conj,SIGUSR1); //Quitamos SIGUSR1 de conj
201                 sigdelset (&conj,SIGTERM);
202                 sa.sa_handler = manejadoraSIGUSR1;
203                 sigemptyset(&sa.sa_mask);
204                 sa.sa_flags=0;
205                 if(sigaction(SIGUSR1, &sa, NULL)==-1) return -1;
206 
207                 switch (f.forkN3 = fork())
208                 {
209                 case -1:
210                     perror("prueba\n");
211                     return -1;
212                 case 0: //N3
213                     printf("Soy N3, PID %d, padre %d\n", getpid(), getppid());
214                     f.forkDestino=f.forkH4;
215                     while (flag == 0)
216                         sigsuspend (&conj);
217                     break;
218                 } //fin switch N3
219                 printf("Soy H3, PID %d, padre %d \n", getpid(), getppid ());
220                 f.forkDestino=f.forkN3;
221                 sa.sa_handler = terminarN3;
222                 if(sigaction(SIGTERM, &sa, NULL)==-1) return -1;
223                 while (flag == 0)
224                     sigsuspend (&conj);
225                 break;
226 
227             default: //Padre
228                 sigdelset (&conj,SIGUSR1);
229                 sa.sa_handler=manejadoraPadre;
230                 sigfillset(&sa.sa_mask);
231                 sa.sa_flags=0;
232                 if(sigaction(SIGUSR1, &sa, NULL)==-1) return -1;
233 
234                 sigdelset (&conj,SIGUSR2);
235                 sa.sa_handler=manejadoraPadre;
236                 sigfillset(&sa.sa_mask);
237                 sa.sa_flags=0;
238                 if(sigaction(SIGUSR2, &sa, NULL)==-1) return -1;
239 
240                 if(sigprocmask(SIG_UNBLOCK,&conj,NULL)==-1);
241                 sigfillset(&sa.sa_mask);
242                 sa.sa_handler=terminarHijos;
243                 sa.sa_flags=0;
244                 if (sigaction(SIGALRM, &sa,NULL)==-1) return -1;
245                 alarm (25);
246 
247                 //se inicia el envio de seniales
248                 if(kill(f.forkH3, SIGUSR1)==-1)
249                     perror("Error");
250 
251                 while (flag == 0){
252                     sigsuspend (&conj);
253                     vueltas ++;
```

```
254                     }
255                     //se hace todo durante 25s y cuando pasan, se salta a terminarHijos
256 
257                 }                   //fin switch H3
258             }
259                                     //fin switch H2
260         }
261                                 //fin switch H4
262     }
263                                  //fin switch
264 
265     return 0;
266 }
267 
268 
269 
270 
271 
272 
273 
274 
275 
276 
```