Segunda práctica
El efecto dominó

Primer miembro: Alex García Alegre (i1307548)
Segundo miembro: Marcos Basanta García (i0960681)

3 de Diciembre de 2024

```
1   #define _HPUX_SOURCE
2
3   #include <stdio.h>
4   #include <stdlib.h>
5   #include <unistd.h>
6   #include <signal.h>
7   #include <sys/wait.h>
8   #include <sys/types.h>
9   #include <fcntl.h>
10  #include <sys/file.h>
11  #include <sys/mman.h>
12
13  #define PID_FILE "pids.dat"
14  #define NUM_PROCESOS 6
15  #define PID_SIZE sizeof(pid_t)
16  #define FILE_SIZE (NUM_PROCESOS * PID_SIZE)
17
18  pid_t pidPrincipal;
19  pid_t *pids;
20  pid_t pidHijo[2];
21
22  void crea_jerarquia();
23  void configurar_manejador();
24  void manejador(int sig);
25  void escribir_pid(pid_t pid, int index);
26  pid_t leer_pid(int index);
27  void proyectar_archivo(int file);
28  void desproyectar_archivo(int file);
29  const char* obtener_cabecera();
30
31  int main() {
32    int i, file;
33    pidPrincipal = getpid();
34
35    puts(obtener_cabecera());
```

```
36      fflush(stdout);
37
38      proyectar_archivo(file);
39      configurar_manejador();
40
41      fprintf(stdout, "Soy el proceso patriarca, con pid %d\n", pidPrincipal);
42      fflush(stdout);
43      crea_jerarquia();
44
45      pause();
46
47      if (getpid() == pidPrincipal) {
48        kill(pidHijo[0], SIGTERM);
49
50        waitpid(pidHijo[0], NULL, 0);
51        fprintf(stdout, "Eliminacion de todo el arbol completada exitosamente\n");
52        fflush(stdout);
53
54        desproyectar_archivo(file);
55      }
56
57      return 0;
58    }
59
60    void crea_jerarquia() {
61      pid_t pidYo;
62      switch(pidHijo[0] = fork()) { // Proceso 38
63        case -1:
64          fprintf(stderr, "main:proceso38:fork");
65          fflush(stdout);
66          return;
67          break;
68        case 0:
69          switch (pidHijo[0] = fork()) { // Proceso 39
70            case -1:
```

```
71                 fprintf(stderr, "main:proceso39:fork");
72                 fflush(stdout);
73                 return;
74                 break;
75             case 0:
76                 switch(pidHijo[0] = fork()) { // Proceso 40
77                     case -1:
78                         fprintf(stderr, "main:proceso40:fork");
79                         fflush(stdout);
80                         return;
81                         break;
82                     case 0:
83                         switch (pidHijo[0] = fork()) { // Proceso 42
84                             case -1:
85                                 fprintf(stderr, "main:proceso42:fork");
86                                 fflush(stdout);
87                                 return;
88                                 break;
89                             case 0:
90                                 switch (pidHijo[0] = fork()) { // Proceso 46
91                                     case -1:
92                                         fprintf(stderr, "main:proceso46:fork");
93                                         fflush(stdout);
94                                         return;
95                                         break;
96                                     case 0:
97                                         switch (pidHijo[0] = fork()) { // Proceso 50
98                                             case -1:
99                                                 fprintf(stderr, "main:proceso50:fork");
100                                                fflush(stdout);
101                                                return;
102                                                break;
103                                            case 0:
104                                                switch (pidHijo[0] = fork()) { // Proceso 54
105                                                    case -1:
```

```
106                     fprintf(stderr, "main:proceso54:fork");
107                     fflush(stdout);
108                     return;
109                     break;
110                   case 0:
111                     pidYo=getpid();
112                     escribir_pid(pidYo, 2);
113                     switch (pidHijo[0] = fork()) { // Proceso 56
114                       case -1:
115                         fprintf(stderr, "main:proceso56:fork");
116                         fflush(stdout);
117                         return;
118                         break;
119                       case 0:
120                         pidYo=getpid();
121                         escribir_pid(pidYo, 4);
122                         switch (pidHijo[0] = fork()) { // Proceso 57
123                           case -1:
124                             fprintf(stderr, "main:proceso57:fork");
125                             fflush(stdout);
126                             return;
127                             break;
128                           case 0:
129                             switch (pidHijo[0] = fork()) { // Proceso 58
130                               case -1:
131                                 fprintf(stderr, "main:proceso58:fork");
132                                 fflush(stdout);
133                                 return;
134                                 break;
135                               case 0:
136                                 pidHijo[0] = -1;
137                                 pidHijo[1] = -1;
138                                 pause();
139                             }
140                             pidHijo[1] = -1;
```

```
141                                     pause();
142                                     break;
143                                 }
144                                 pidHijo[1] = -1;
145                                 pause();
146                                 break;
147                             }
148                             pidHijo[0] = -1;
149                             pidHijo[1] = -1;
150                             pause();
151                             break;
152                         }
153                         pidHijo[0] = -1;
154                         pidHijo[1] = -1;
155                         pause();
156                         break;
157                     }
158                     pidHijo[1] = -1;
159                     pause();
160                     break;
161                 }
162                 pidHijo[1] = -1;
163                 pause();
164                 break;
165             }
166             switch (pidHijo[1] = fork()) { // Proceso 43
167               case -1:
168                 fprintf(stderr, "main:proceso43:fork");
169                 fflush(stdout);
170                 return;
171                 break;
172               case 0:
173                 switch (pidHijo[0] = fork()) { // Proceso 47
174                   case -1:
175                     fprintf(stderr, "main:proceso47:fork");
```

```
176                         fflush(stdout);
177                         return;
178                         break;
179                       case 0:
180                         switch (pidHijo[0] = fork()) { // Proceso 51
181                           case -1:
182                             fprintf(stderr, "main:proceso51:fork");
183                             fflush(stdout);
184                             return;
185                             break;
186                           case 0:
187                             pidYo=getpid();
188                             escribir_pid(pidYo, 0);
189                             pidHijo[0] = -1;
190                             pidHijo[1] = -1;
191                             pause();
192                             break;
193                         }
194                         pidHijo[1] = -1;
195                         pause();
196                         break;
197                     }
198                     pidHijo[1] = -1;
199                     pause();
200                     break;
201                 }
202                 pause();
203                 break;
204             }
205             switch (pidHijo[1] = fork()) { // Proceso 41
206               case -1:
207                 fprintf(stderr, "main:proceso41:fork");
208                 fflush(stdout);
209                 return;
210                 break;
```

```
211                 case 0:
212                   switch (pidHijo[0] = fork()) { // Proceso 44
213                     case -1:
214                       fprintf(stderr, "main:proceso44:fork");
215                       fflush(stdout);
216                       return;
217                       break;
218                     case 0:
219                       switch (pidHijo[0] = fork()) { // Proceso 48
220                         case -1:
221                           fprintf(stderr, "main:proceso48:fork");
222                           fflush(stdout);
223                           return;
224                           break;
225                         case 0:
226                           switch (pidHijo[0] = fork()) { // Proceso 52
227                             case -1:
228                               fprintf(stderr, "main:proceso52:fork");
229                               fflush(stdout);
230                               return;
231                               break;
232                             case 0:
233                               switch (pidHijo[0] = fork()) { // Proceso 55
234                                 case -1:
235                                   fprintf(stderr, "main:proceso55:fork");
236                                   fflush(stdout);
237                                   return;
238                                   break;
239                                 case 0:
240                                   pidYo=getpid();
241                                   escribir_pid(pidYo, 3);
242                                   pidHijo[0] = -1;
243                                   pidHijo[1] = -1;
244                                   pause();
245                                   break;
```

```
246                         }
247                         pidHijo[0] = -1;
248                         pidHijo[1] = -1;
249                         pause();
250                         break;
251                     }
252                     pidHijo[1] = -1;
253                     pause();
254                     break;
255                 }
256                 pidHijo[1] = -1;
257                 pause();
258                 break;
259             }
260             switch (pidHijo[1] = fork()) { // Proceso 45
261               case -1:
262                 fprintf(stderr, "main:proceso45:fork");
263                 fflush(stdout);
264                 return;
265                 break;
266               case 0:
267                 switch (pidHijo[0] = fork()) { // Proceso 49
268                   case -1:
269                     fprintf(stderr, "main:proceso49:fork");
270                     fflush(stdout);
271                     return;
272                     break;
273                   case 0:
274                     switch (pidHijo[0] = fork()) { // Proceso 53
275                       case -1:
276                         fprintf(stderr, "main:proceso53:fork");
277                         fflush(stdout);
278                         return;
279                         break;
280                       case 0:
```

```
281                                 pidYo=getpid();
282                                 escribir_pid(pidYo, 1);
283                                 pidHijo[0] = -1;
284                                 pidHijo[1] = -1;
285                                 pause();
286                                 break;
287                             }
288                             pidHijo[1] = -1;
289                             pause();
290                             break;
291                         }
292                         pidHijo[1] = -1;
293                         pause();
294                         break;
295                     }
296                     pause();
297                     break;
298                 }
299                 pause();
300                 break;
301             }
302             pidHijo[1] = -1;
303             pause();
304             break;
305         default:
306             pidHijo[1] = -1;
307             break;
308     }
309 }
310
311 void configurar_manejador() {
312     struct sigaction sa;
313     // Configurar la estructura sigaction
314     sa.sa_handler = manejador;
315     sigemptyset(&sa.sa_mask);
```

```
316         sa.sa_flags = 0;
317         // Establecer el manejador de señales para SIGTERM
318         if (sigaction(SIGTERM, &sa, NULL) == -1) {
319             fprintf(stderr, "Error al establecer el manejador de señales");
320             fflush(stdout);
321             exit(EXIT_FAILURE);
322         }
323     }
324
325     void manejador(int sig) {
326         pid_t pidYo;
327         pidYo = getpid();
328         if (sig == SIGTERM) {
329             if(pidYo == pidPrincipal) {
330                 fprintf(stdout, "Comienzo de la propagacion de la señal...\n");
331                 fflush(stdout);
332                 return;
333             }
334             if (pidYo == leer_pid(0)) { // Proceso 51
335                 pidHijo[0] = leer_pid(2); // Proceso 54
336                 pidHijo[1] = -1;
337             } else if(pidYo == leer_pid(1)) { // Proceso 53
338                 pidHijo[0] = leer_pid(3); // Proceso 55
339                 pidHijo[1] = -1;
340             } else if (pidYo == leer_pid(3)) { // Proceso 55
341                 pidHijo[0] = leer_pid(4); // Proceso 56
342                 pidHijo[1] = -1;
343             }
344             if(pidHijo[0] != -1) {
345                 kill(pidHijo[0], SIGTERM);
346             }
347             if (pidHijo[1] != -1) {
348                 kill(pidHijo[1], SIGTERM);
349             }
350             if(pidHijo[0] != -1) {
```

```
351             waitpid(pidHijo[0], NULL, 0);
352         }
353         if (pidHijo[1] != -1) {
354             waitpid(pidHijo[1], NULL, 0);
355         }
356         exit(0);
357       }
358     }
359
360     void escribir_pid(pid_t pid, int index) {
361       pids[index] = pid;
362     }
363
364     pid_t leer_pid(int index) {
365       return pids[index];
366     }
367
368     void proyectar_archivo(int file) {
369       int fd = open(PID_FILE, O_CREAT | O_RDWR, 0666);
370
371       if (fd == -1) {
372         fprintf(stderr, "Error al crear el archivo");
373         fflush(stdout);
374         exit(EXIT_FAILURE);
375       }
376       if (ftruncate(fd, FILE_SIZE) == -1) {
377         fprintf(stderr, "Error al ajustar el tamaño del archivo");
378         fflush(stdout);
379         exit(EXIT_FAILURE);
380       }
381
382       pids = (pid_t *)mmap(0, FILE_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
383
384       if (pids == MAP_FAILED) {
385         fprintf(stderr, "Error al mapear el archivo");
```

```
386             fflush(stdout);
387             exit(EXIT_FAILURE);
388         }
389
390         close(fd);
391     }
392
393     void desproyectar_archivo(int file) {
394         // Desmapear la memoria y cerrar el archivo
395         if (munmap((void*)pids, NUM_PROCESOS * PID_SIZE) == -1) {
396             fprintf(stderr, "Error en munmap");
397             fflush(stdout);
398         }
399
400         close(file);
401
402         if (remove(PID_FILE) == 0) {
403             fprintf(stdout, "Archivo de PIDs eliminado correctamente.\n");
404             fflush(stdout);
405         } else {
406             fprintf(stderr, "Error al eliminar el archivo de PIDs\n");
407             fflush(stdout);
408         }
409     }
410
411     const char* obtener_cabecera() {
412         static const char cabecera[] =
413             "\n"
414             "╔═══════════════════════════════════════════════════════════════════════════════╗\n"
415             "║          PROGRAMA DE ÁRBOL DE PROCESOS                ║\n"
416             "╚═══════════════════════════════════════════════════════════════════════════════╝\n"
417             "\n"
418             "Estructura del árbol de procesos:\n\n"
419             "               37           \n"
420             "               |            \n"
```

```
421             "              38          \n"
422             "              |           \n"
423             "              39          \n"
424             "            /    \\        \n"
425             "           40     41       \n"
426             "          / \\   /  \\     \n"
427             "        42  43  44   45    \n"
428             "        |   |   |    |     \n"
429             "        46  47  48   49    \n"
430             "        |   |   |    |     \n"
431             "        50  51  52   53    \n"
432             "          \\       \\      \n"
433             "           54      55      \n"
434             "             \\            \n"
435             "              56          \n"
436             "              |           \n"
437             "              57          \n"
438             "              |           \n"
439             "              58        \n\n"
440         "Iniciando creación de procesos...\n";
441
442     return cabecera;
443 }
```