

1. Feladat :

. Készítsen C nyelvű programot, ahol egy szülő processz létrehoz egy csővezetékét, a gyerek processz beleír egy szöveget a csővezetékbe (A kiírt szöveg: XY neptunkod), a szülő processz ezt kiolvassa, és kiírja a standard kimenetre. Mentés: neptunkod\_unnamed.c

```
Start here x nxych1_named.c x nxych1_unnamed.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <sys/wait.h>
4  #include <unistd.h>
5  #include <string.h>
6
7  #define SIZE 64
8
9  int main() {
10     int _pipe[2];
11     int _pid;
12     char _buffer[SIZE];
13
14     if (pipe(_pipe) == -1) { // pipe hiba kezeles
15         perror("pipe() Error");
16         exit(-1);
17     }
18
19     _pid = fork();
20     if (_pid == -1) { // fork hiba kezeles
21         perror("fork() Error");
22         exit(-1);
23     }
24
25     if (_pid == 0) { // gyerek
26         printf("%d: I'm a child\n", getpid());
27
28         close(_pipe[0]); // lezaras mivel beolvas
29
30         strcpy(_buffer, "Kiss Mate NXYCH1\0");
31         write(_pipe[1], _buffer, sizeof(_buffer)); // szoveg iras
32
33         close(_pipe[1]); // lezaras
34     } else { // szulo
35         printf("%d: I'm a parent\n", getpid());
36
37         close(_pipe[1]); // lezaras mivel olvas
38
39         while (read(_pipe[0], _buffer, sizeof(_buffer)) > 0) { // olvas
40             printf("%s", _buffer);
41         }
42         printf("\n");
43
44         close(_pipe[0]); // lezaras
45     }
46     wait(NULL);
47     return 0;
48 }
49
```

2. Feladat: .

Készítsen C nyelvű programot, ahol egy szülő processz létrehoz egy nevesített csővezetékét (neve: neptunkod), a gyerek processz beleír egy szöveget a csővezetékbe (A hallgató neve: pl. Keserű Ottó), a szülő processz ezt kiolvassa, és kiírja a standard kimenetre. Mentés: neptunkod\_named.c

```
Start here x nxych1_named.c x nxych1_unnamed.c x
1  #include <stdio.h>
2  #include <sys/file.h>
3  #include <sys/types.h>
4  #include <sys/stat.h>
5  #include <fcntl.h>
6  #include <signal.h>
7  #include <sys/wait.h>
8  #include <stdlib.h>
9  #include <string.h>
10 #include <unistd.h>
11
12 #define NAME "NXYCH1"
13 #define SIZE 64
14
15 int main() {
16     pid_t _pid;
17     int _fifo;
18     int _open;
19     char _buffer[SIZE];
20     int _temp;
21
22     _pid = fork();
23     if (_pid == -1) {
24         perror("fork() Error");
25         exit(-1);
26     }
27
28     if (_pid == 0) { // a child
29         printf("%d: I'm a child\n", getpid());
30
31         _open = open(NAME, O_RDWR);
32         if (_open == -1) {
33             perror("open() Error");
34             exit(-1);
35         }
36
37         strcpy(_buffer, "Kiss Mate NXYCH1\0");
38         write(_open, _buffer, SIZE);
39     }
```

```
Start here x nxych1_named.c x nxych1_unnamed.c x
39
40     close(_open);
41     exit(1);
42 } else { // a parent
43     printf("%d: I'm a parent [1.]\n", getpid());
44
45     _fifo = mkfifo(NAME, 0100);
46     if (_fifo == -1) {
47         perror("mkfifo() Error");
48         exit(-1);
49     }
50     close(_fifo);
51     wait(NULL);
52
53     printf("%d: I'm a parent [2.]\n", getpid());
54
55     _open = open(NAME, O_RDWR);
56     if (_open == -1) {
57         perror("open()");
58         exit(-1);
59     }
60
61     printf("0\n");
62
63     while (read(_fifo, _buffer, SIZE) > 0) {
64         printf("%s", _buffer); // aaaaaa
65     }
66     printf("\n");
67
68     close(_open);
69     unlink(NAME);
70
71     exit(1);
72 }
73
74 return 0;
75
76 }
77
```

### 3. Feladat:

Írjon C nyelvű programot, amelyik kill() seg.-vel SIGALRM-et küld egy argumentumként megadott PID-u processznek, egy másik futó program a SIGALRM-hez rendeljen egy fv.-t amely kiírja pl.neptunkodot, továbbá pause() fv.-el blokkolódjon, majd kibillenés után jelezze, hogy kibillent és terminálódjon. Mentés. neptunkod\_gyak9\_3.c

```
Start here x nxych1_gyak_3.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <signal.h>
5
6  int kill(pid_t pid, int sig);
7
8  int main(int argc, char **argv)
9  {
10     if(argc != 2)
11     {
12         printf("Hasznalat: ./nxych1_gyak9_3 PID\n");
13         return 1;
14     }
15
16     pid_t pid = (pid_t)atoi(argv[1]);
17     kill(pid, SIGALRM);
18     return 0;
19 }
20
```

```
Start here x nxych1_gyak_3_2.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <signal.h>
5
6  void AlarmHandler(int sig);
7
8  int main(void)
9  {
10     if (signal(SIGALRM, AlarmHandler) == SIG_ERR)
11     {
12         printf("Nem sikerult handlert allitani a(z) \"SIGALRM\" jelre!\n");
13         return 1;
14     }
15
16     pause();
17     return 0;
18 }
19
20 void AlarmHandler(int sig)
21 {
22     printf("NXYCH1\n Mostantol nem blokkol !\n");
23     exit(1);
24 }
25
```

#### 4. Feladat:

Írjon C nyelvű programot, amelyik a SIGTERM-hez hozzárendel egy fv-t., amelyik kiírja az int paraméter értékét, majd végtelen ciklusban fusson, 3 sec-ig állandóan blokkolódva elindítás után egy másik shell-ben kill paranccsal (SIGTERM) próbálja terminálni, majd SIGKILL-el.

Mentés. neptunkod\_gyak9\_4.c

```
Start here x nxych1_gyak_4.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #include <unistd.h>
5  #include <signal.h>
6
7  void TerminationHandler(int sig);
8
9  int main(void)
10 {
11     if (signal(SIGTERM, TerminationHandler) == SIG_ERR)
12     {
13         printf("Nem sikerult a handlert allitani a \"SIGTERM\" jelre!\n");
14         return 0;
15     }
16
17     while(1)
18     {
19         printf("Varakozom...\n");
20         sleep(3);
21     }
22
23     return 0;
24 }
25
26 void TerminationHandler(int sig)
27 {
28     signal(sig, SIG_IGN);
29     printf("SIGTERM signal: %d\n", sig);
30 }
31
```