

*Operációs rendszerek*  
*BSc*  
*2. Gyak. 2022. 04. 07.*

*Készítette: Dobra Gábor*  
*Bsc Mérnökinformatikus*  
*XQBTIW*

*Miskolc, 2022*

## 1. Feladat:

A programban a `system( )` rendszerhívás érvényes parancsot futtat.

```
borde@DESKTOP-B61GM0V:/mnt/c/prog$ gcc -o 1fel 1fel.c
borde@DESKTOP-B61GM0V:/mnt/c/prog$ ./1fel
1fel 1fel.c a.txt b.txt c.txt
Visszatérési érték: 0
sh: 1: cm: not found
Visszatérési érték: 32512
borde@DESKTOP-B61GM0V:/mnt/c/prog$
```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main() {
5     int return_value = system("ls");
6     printf("Visszatérési érték: %d\n", return_value);
7
8     return_value = system("cm");
9     printf("Visszatérési érték: %d\n", return_value);
10
11     return 0;
12 }
```

## 2. Feladat:

A következő képernyőfotón az látszik, ahogy az UNIX a parancsokat bekéri és végrehajtsa.

```
borde@DESKTOP-B61GM0V:/mnt/c/prog$ gcc -o 2fel 2fel.c
```

```
borde@DESKTOP-B61GM0V:/mnt/c/prog$ ./2fel
```

Adja meg a végrehajtandó parancsot!

```
pwd
```

A pwd parancs eredménye:

```
/mnt/c/prog
```

Adja meg a végrehajtandó parancsot!

```
date
```

A date parancs eredménye:

```
Sun Apr  3 10:27:15 CEST 2022
```

Adja meg a végrehajtandó parancsot!

```
who
```

A who parancs eredménye:

Adja meg a végrehajtandó parancsot!

```
whois
```

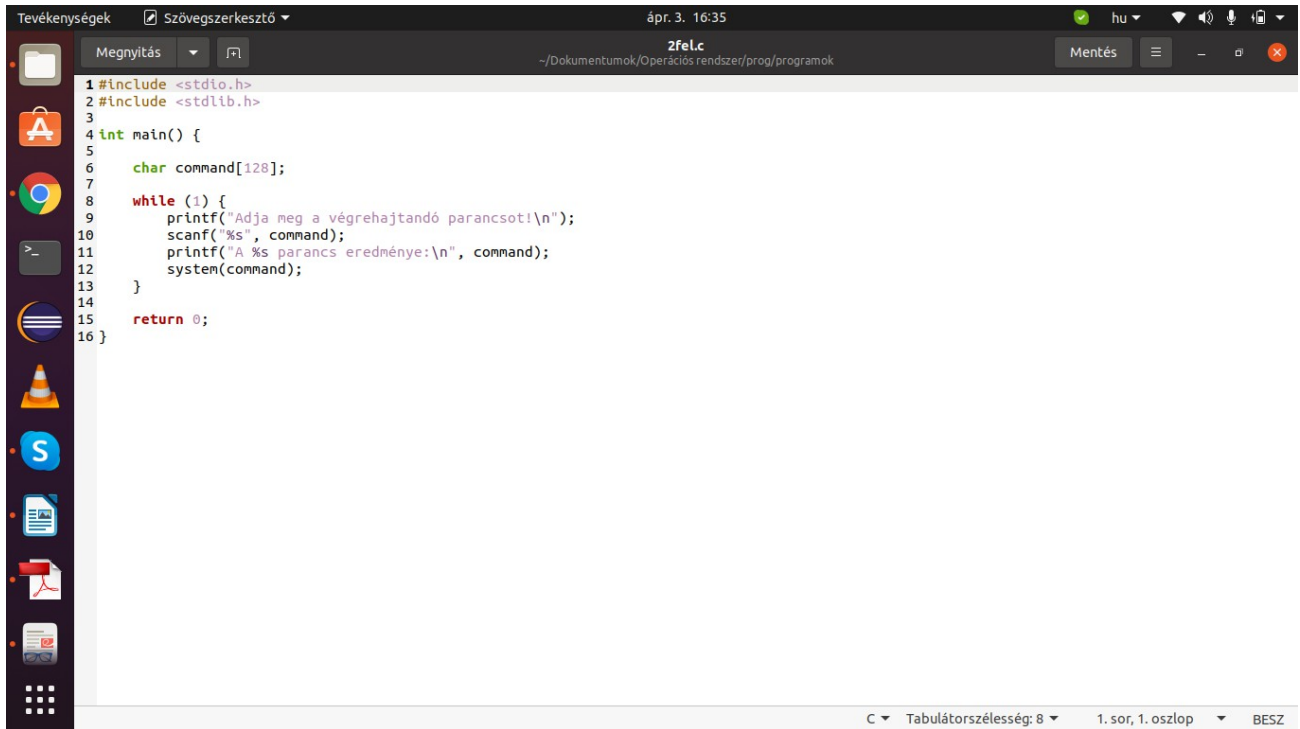
A whois parancs eredménye:

```
sh: 1: whois: not found
```

Adja meg a végrehajtandó parancsot!

```
^C
```

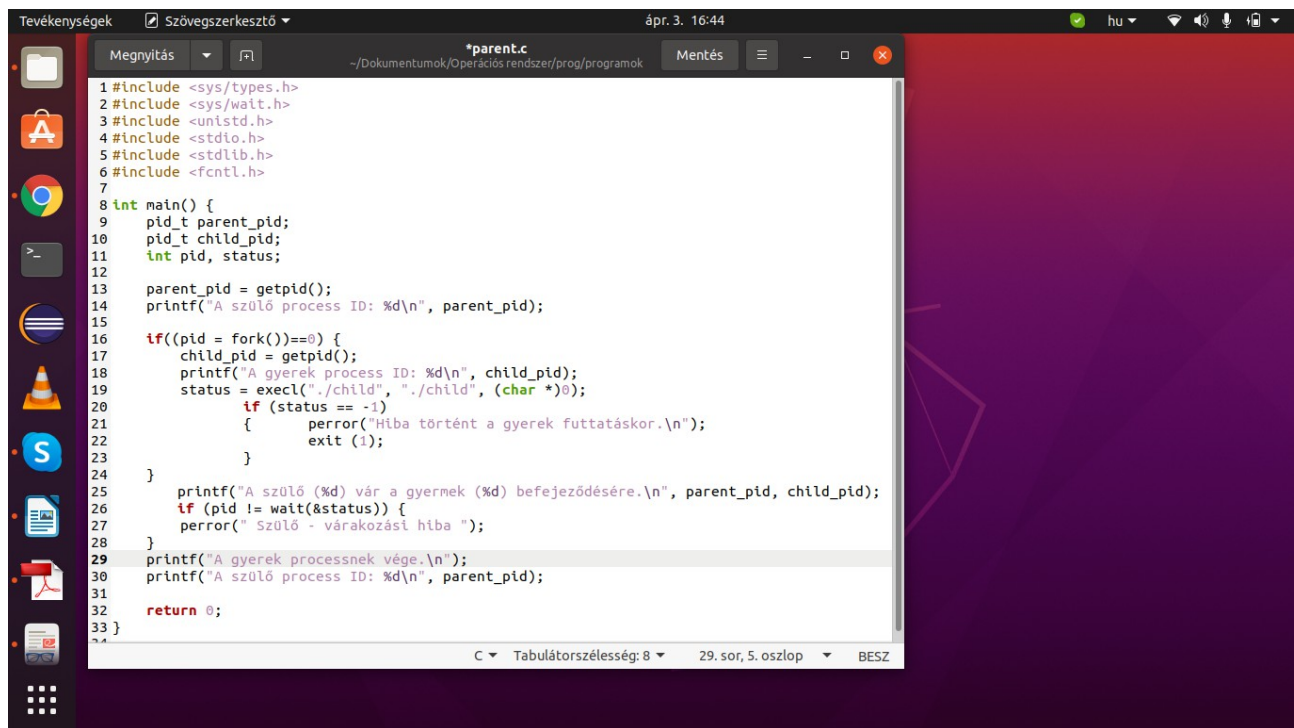
```
borde@DESKTOP-B61GM0V:/mnt/c/prog$
```



```
1#include <stdio.h>
2#include <stdlib.h>
3
4int main() {
5
6    char command[128];
7
8    while (1) {
9        printf("Adj meg a végrehajtandó parancsot!\n");
10       scanf("%s", command);
11       printf("A %s parancs eredménye:\n", command);
12       system(command);
13   }
14
15   return 0;
16 }
```

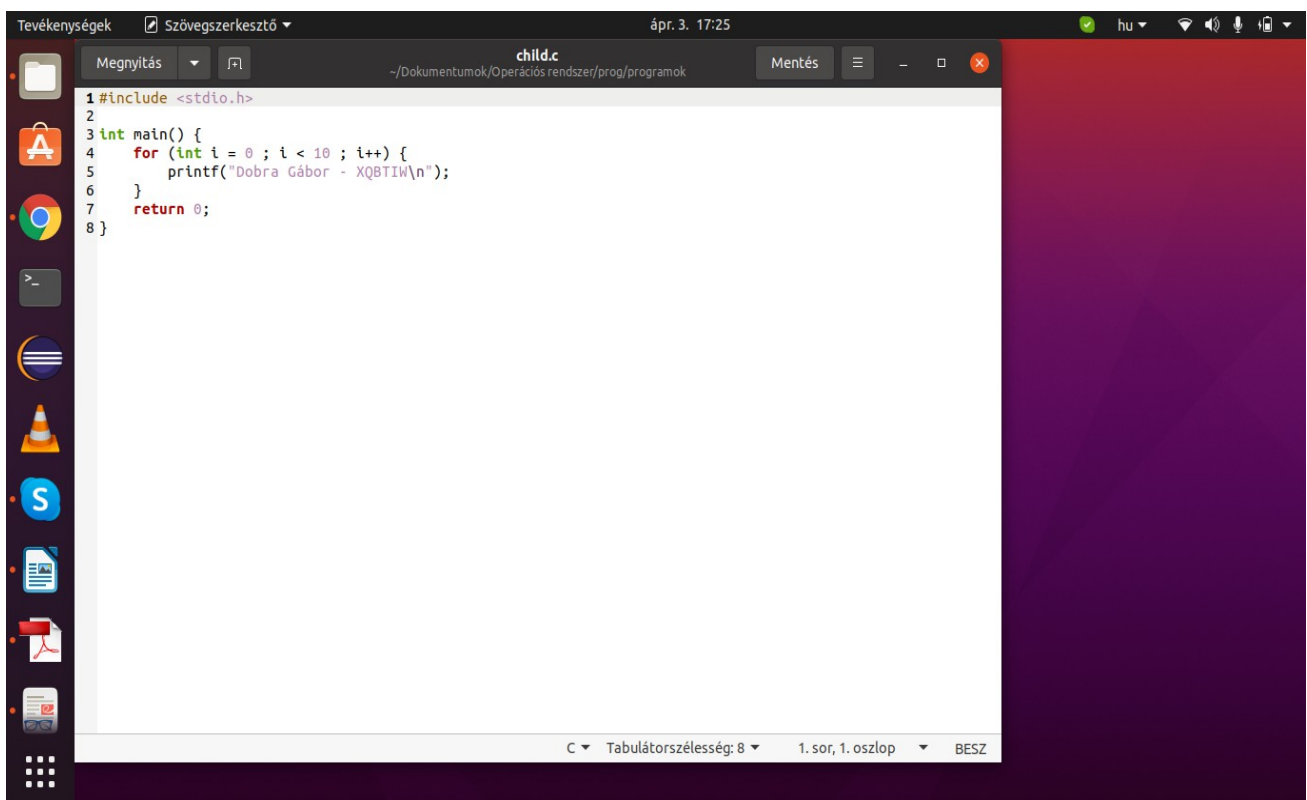
### 3. Feladat:

A parent.c elindít egy gyermek processzt, ami különbözik a szülőtől. A szülő megvárja a gyermek lefutását. A gyermek szöveget ír a szabványos kimenetre.



The screenshot shows a Linux desktop with a dark theme. A text editor window titled "parent.c" is open, displaying C code for a parent process. The code includes headers for `<sys/types.h>`, `<sys/wait.h>`, `<unistd.h>`, `<stdio.h>`, `<stdlib.h>`, and `<fcntl.h>`. The `main` function gets the parent PID, forks a child process, prints both PIDs, and waits for the child to finish. The status bar at the bottom indicates "C", "Tabulátorszélesség: 8", "29. sor, 5. oszlop", and "BESZ".

```
1#include <sys/types.h>
2#include <sys/wait.h>
3#include <unistd.h>
4#include <stdio.h>
5#include <stdlib.h>
6#include <fcntl.h>
7
8int main() {
9    pid_t parent_pid;
10   pid_t child_pid;
11   int pid, status;
12
13   parent_pid = getpid();
14   printf("A szülő process ID: %d\n", parent_pid);
15
16   if((pid = fork())==0) {
17       child_pid = getpid();
18       printf("A gyerek process ID: %d\n", child_pid);
19       status = execl("./child", "./child", (char *)0);
20       if (status == -1)
21           perror("Hiba történt a gyerek futtatáskor.\n");
22       exit(1);
23   }
24
25   printf("A szülő (%d) vár a gyermek (%d) befejeződésére.\n", parent_pid, child_pid);
26   if (pid != wait(&status)) {
27       perror("Szülő - várákozási hiba ");
28   }
29   printf("A gyerek processnek vége.\n");
30   printf("A szülő process ID: %d\n", parent_pid);
31
32   return 0;
33 }
```



The screenshot shows the same Linux desktop environment, but now the text editor window is titled "child.c". The code is a simple `main` function that loops from 0 to 10, printing "Dobra Gábor - XQBtIW\n" in each iteration. The status bar at the bottom indicates "C", "Tabulátorszélesség: 8", "1. sor, 1. oszlop", and "BESZ".

```
1#include <stdio.h>
2
3int main() {
4   for (int i = 0 ; i < 10 ; i++) {
5       printf("Dobra Gábor - XQBtIW\n");
6   }
7   return 0;
8 }
```

Szabványos kimenet:

```

borde@DESKTOP-B61GM0V:/mnt/c/prog$ gcc -o child child.c
borde@DESKTOP-B61GM0V:/mnt/c/prog$ gcc -o parent parent.c
borde@DESKTOP-B61GM0V:/mnt/c/prog$ ./parent
A szülő process ID: 555
A szülő (555) vár a gyermek (32748) befejeződésére.
A gyerek process ID: 556
Dobra Gábor - XQBTIW
Dobra Gábor - XQBTIW
Dobra Gábor - XQBTIW
Dobra Gábor - XQBTIW
Dobra Gábor - XQBTIW
Dobra Gábor - XQBTIW
Dobra Gábor - XQBTIW
Dobra Gábor - XQBTIW
Dobra Gábor - XQBTIW
Dobra Gábor - XQBTIW
A gyerek processnek vége.
A szülő process ID: 555
borde@DESKTOP-B61GM0V:/mnt/c/prog$

```

#### 4. Feladat:

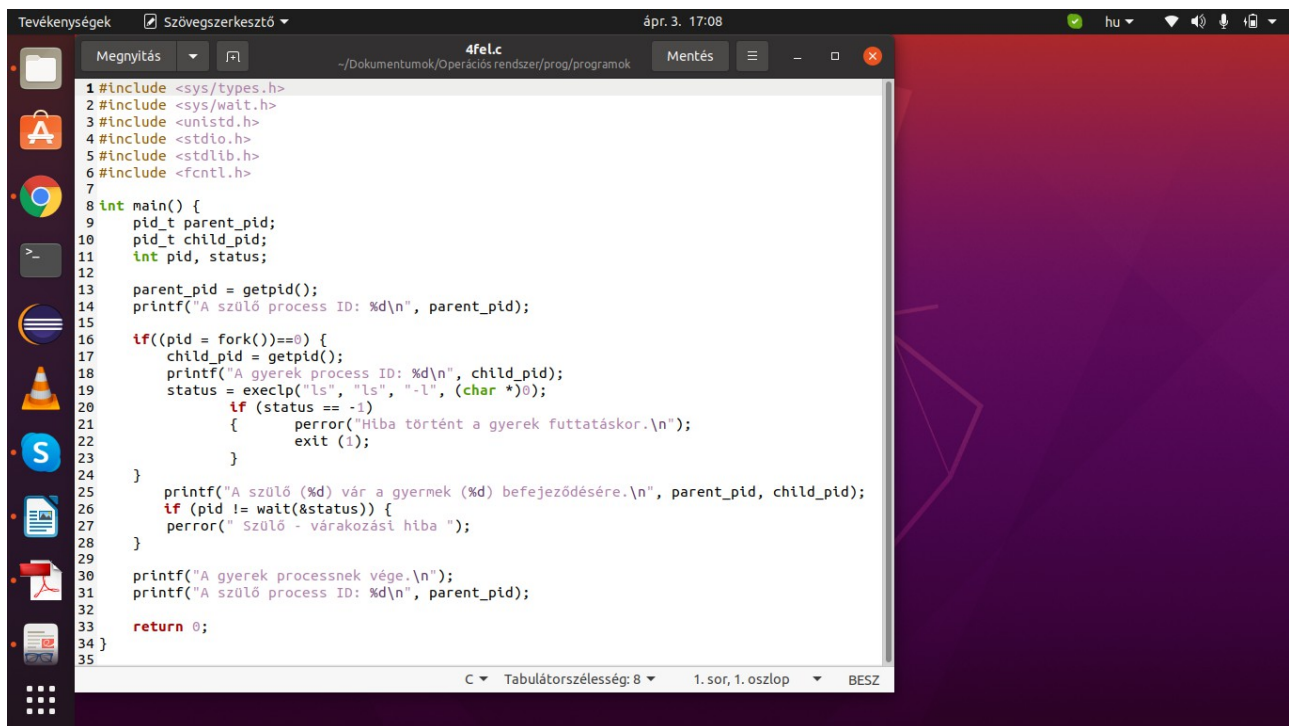
Unix-ban a processz kreálás alapvetően a fork()-kal, vagy a fork()/exec()  
"villával" történik.

Alapvető módszer új processz készítésére. A fork a koncepciója szerint a processz instrukció folyamatát két, konkurrens instrukciófolyammá osztja. A Unix-ban gyermek processz kreálással valósítja meg.

```

borde@DESKTOP-B61GM0V:/mnt/c/prog$ gcc -o 4fel 4fel.c
borde@DESKTOP-B61GM0V:/mnt/c/prog$ ./4fel
A szülő process ID: 576
A szülő (576) vár a gyermek (32560) befejeződésére.
A gyerek process ID: 577
total 108
-rwxrwxrwx 1 borde borde 16744 Apr  3 10:22 1fel
-rwxrwxrwx 1 borde borde   271 Apr  3 10:20 1fel.c
-rwxrwxrwx 1 borde borde 16832 Apr  3 10:26 2fel
-rwxrwxrwx 1 borde borde   300 Apr  3 10:26 2fel.c
-rwxrwxrwx 1 borde borde 17048 Apr  3 11:14 4fel
-rwxrwxrwx 1 borde borde   889 Apr  3 11:13 4fel.c
-rwxrwxrwx 1 borde borde    0 Apr  3 10:15 a.txt
-rwxrwxrwx 1 borde borde    0 Apr  3 10:15 b.txt
-rwxrwxrwx 1 borde borde    0 Apr  3 10:15 c.txt
-rwxrwxrwx 1 borde borde 16696 Apr  3 11:08 child
-rwxrwxrwx 1 borde borde   141 Apr  3 10:43 child.c
-rwxrwxrwx 1 borde borde 17048 Apr  3 11:08 parent
-rwxrwxrwx 1 borde borde   892 Apr  3 11:08 parent.c
A gyerek processnek vége.
A szülő process ID: 576
borde@DESKTOP-B61GM0V:/mnt/c/prog$

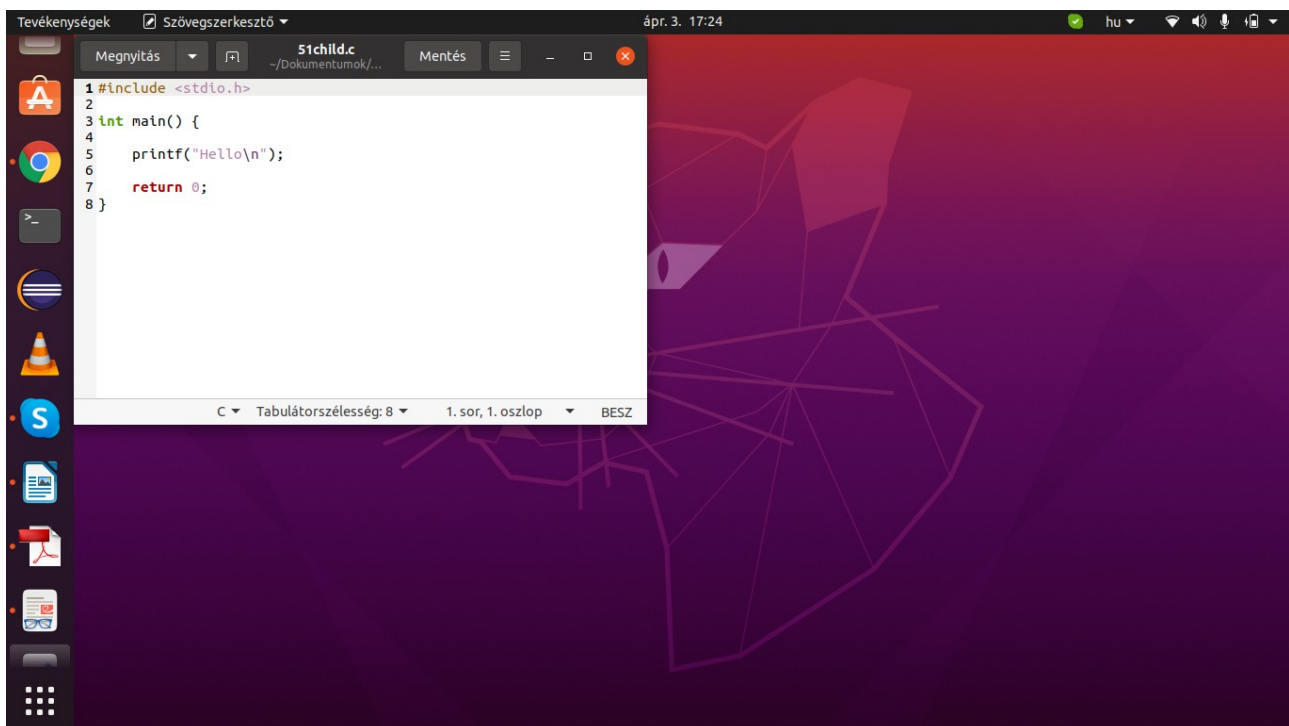
```



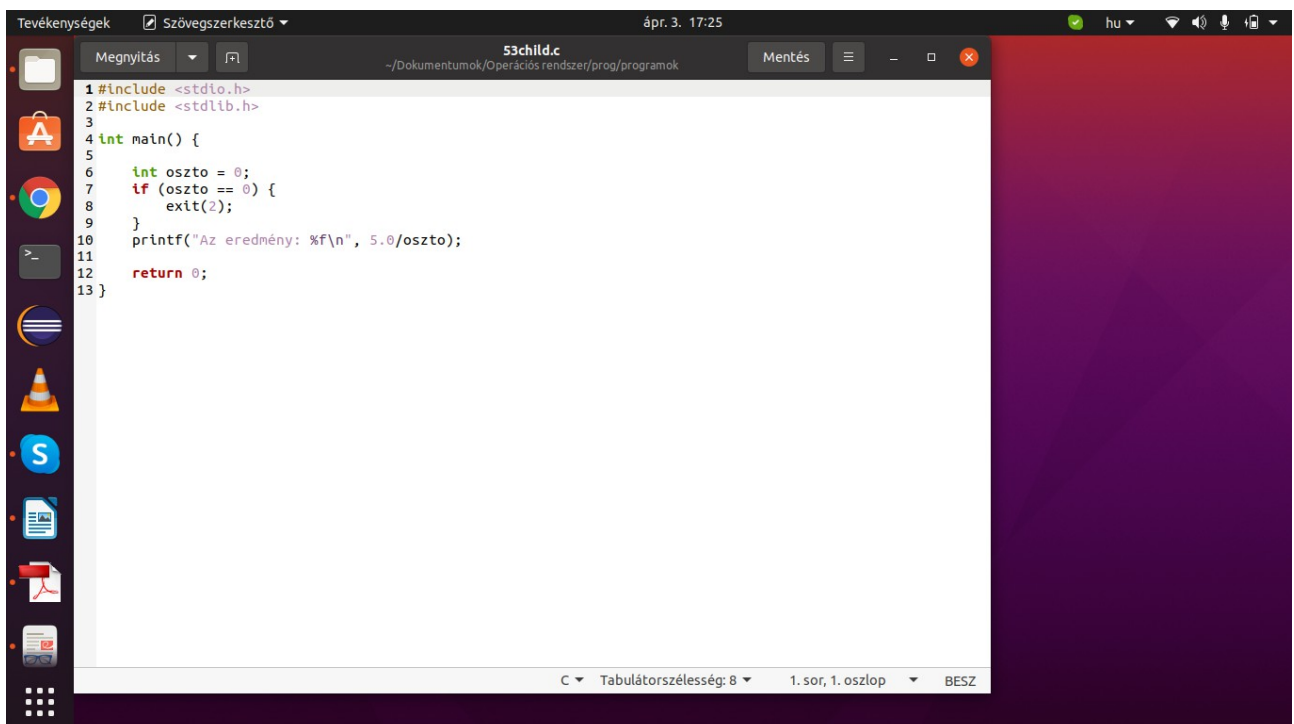
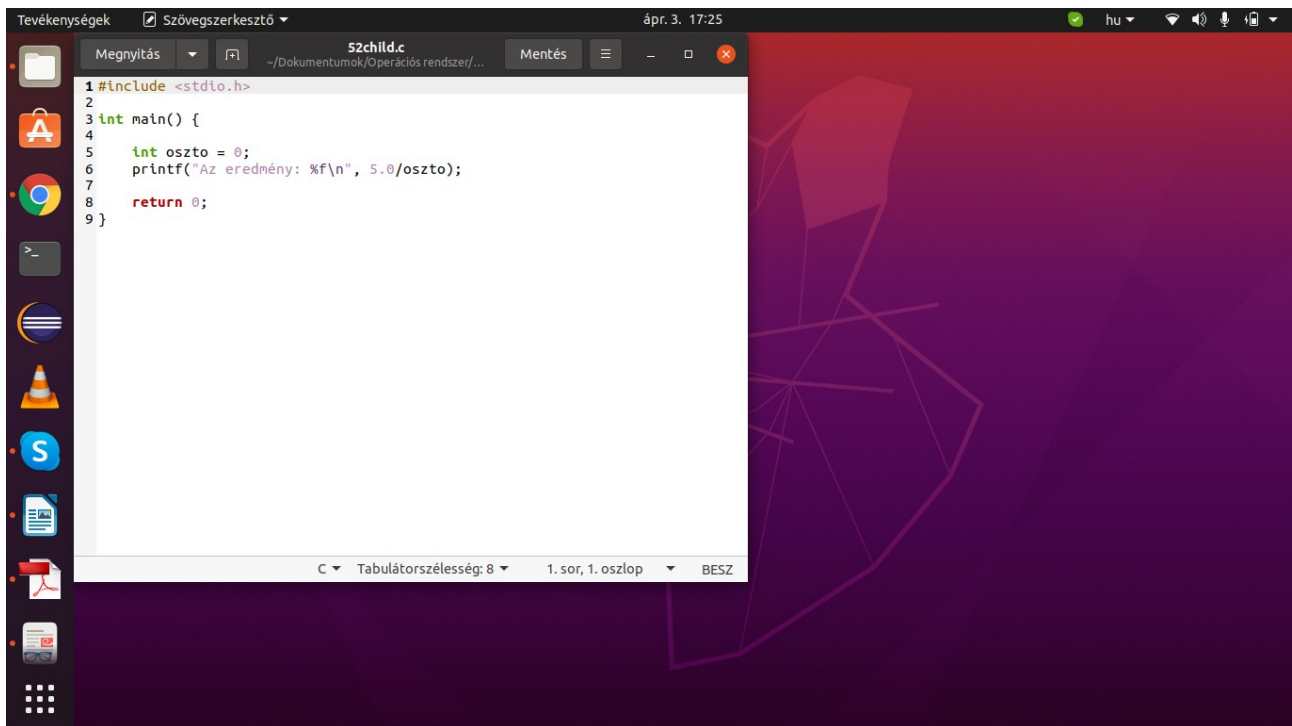
```
1#include <sys/types.h>
2#include <sys/wait.h>
3#include <unistd.h>
4#include <stdio.h>
5#include <stdlib.h>
6#include <fcntl.h>
7
8int main() {
9    pid_t parent_pid;
10   pid_t child_pid;
11   int pid, status;
12
13   parent_pid = getpid();
14   printf("A szülő process ID: %d\n", parent_pid);
15
16   if((pid = fork())==0) {
17       child_pid = getpid();
18       printf("A gyerek process ID: %d\n", child_pid);
19       status = execlp("ls", "ls", "-l", (char *)0);
20       if (status == -1)
21       {
22           perror("Hiba történt a gyerek futtatáskor.\n");
23           exit (1);
24       }
25       printf("A szülő (%d) vár a gyermek (%d) befejeződésére.\n", parent_pid, child_pid);
26       if (pid != wait(&status)) {
27           perror(" Szülő - várakozási hiba ");
28       }
29
30       printf("A gyerek processnek vége.\n");
31       printf("A szülő process ID: %d\n", parent_pid);
32
33       return 0;
34   }
35 }
```

## 5. Feladat:

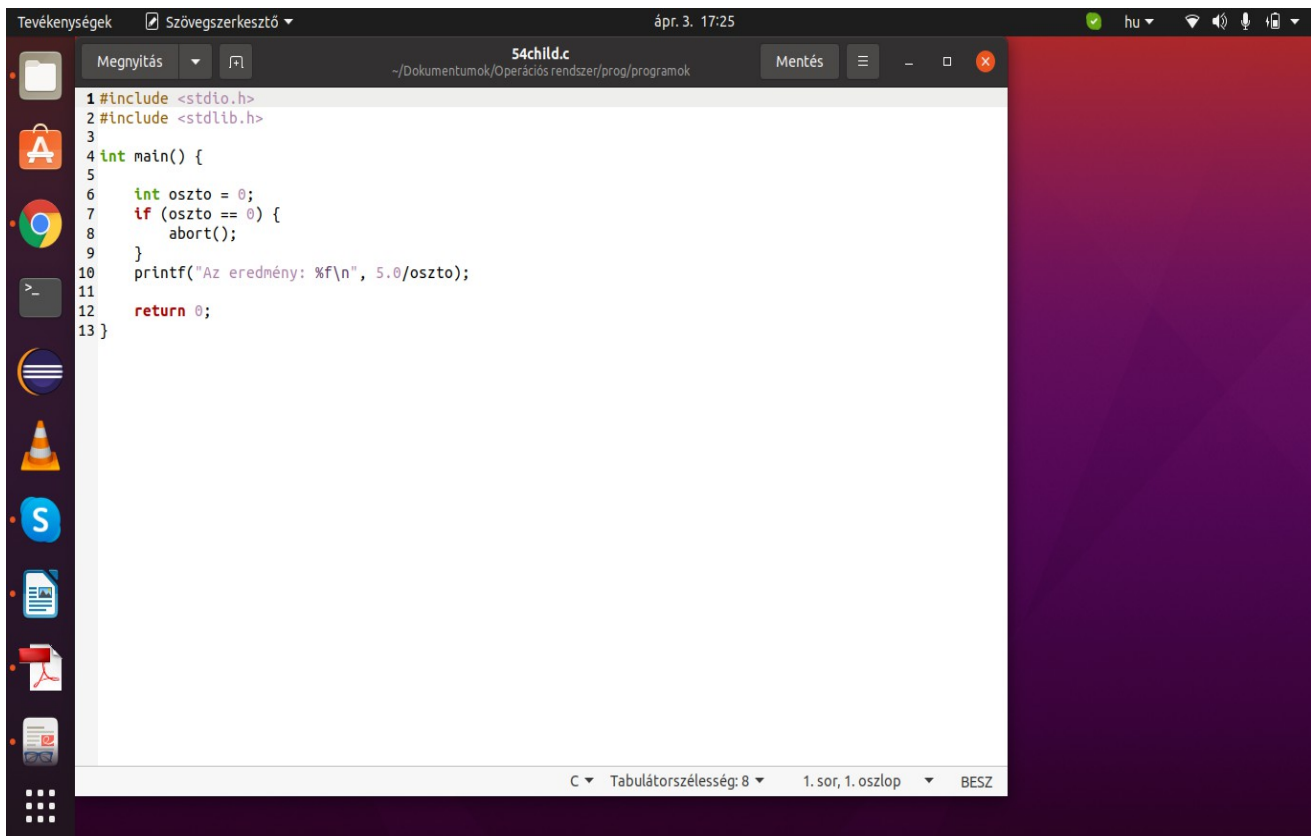
A fork() rendszerhívással hoztam létre gyereket és tanulmányoztam a következő állapotokat: exit, abort, nullával való osztás.



```
1#include <stdio.h>
2
3int main() {
4
5    printf("Hello\n");
6
7    return 0;
8}
```







Az eredmény:

```
borde@DESKTOP-B61GM0V:/mnt/c/prog$ gcc -o 51child 51child.c
borde@DESKTOP-B61GM0V:/mnt/c/prog$ gcc -o 52child 52child.c
borde@DESKTOP-B61GM0V:/mnt/c/prog$ gcc -o 53child 53child.c
borde@DESKTOP-B61GM0V:/mnt/c/prog$ gcc -o 54child 54child.c
borde@DESKTOP-B61GM0V:/mnt/c/prog$ gcc -o 5fel 5fel.c
borde@DESKTOP-B61GM0V:/mnt/c/prog$ ./5fel ./51child ./51child
A szülő process ID: 713
A szülő (713) vár a gyermek (32705) befejeződésére.
A gyerek process ID: 714
Hello
A gyerek státusz: 0
A gyerek processnek vége.
A szülő process ID: 713
borde@DESKTOP-B61GM0V:/mnt/c/prog$ ./5fel ./52child ./52child
A szülő process ID: 715
A szülő (715) vár a gyermek (32721) befejeződésére.
A gyerek process ID: 716
Az eredmény: inf
A gyerek státusz: 0
A gyerek processnek vége.
A szülő process ID: 715
borde@DESKTOP-B61GM0V:/mnt/c/prog$ ./5fel ./53child ./53child
A szülő process ID: 717
A szülő (717) vár a gyermek (32591) befejeződésére.
A gyerek process ID: 718
A gyerek státusz: 512
A gyerek processnek vége.
A szülő process ID: 717
borde@DESKTOP-B61GM0V:/mnt/c/prog$ ./5fel ./54child ./54child
A szülő process ID: 719
A szülő (719) vár a gyermek (32672) befejeződésére.
A gyerek process ID: 720
A gyerek státusz: 134
A gyerek processnek vége.
A szülő process ID: 719
borde@DESKTOP-B61GM0V:/mnt/c/prog$
```

## 6.Feladat:

**Határozza meg FCFS és SJF esetén:**

a.) A befejezési időt?

b.) A várakozási/átlagos várakozási időt?

c.) Ábrázolja Gantt diagram segítségével az aktív/várakozó processzek futásának menetét.

**FCFS:**

FCFS	Érkezés	CPU idő
P1	0	3
P2	1	8
P3	3	2
P4	9	20
P5	12	5

**FCFS megoldás:**

FCFS	Érkezés	CPU idő	Indulás	Befejezés	Várakozás
P1	0	3	0	3	0
P2	1	8	3	11	2
P3	3	2	11	13	8
P4	9	20	13	33	4
P5	12	5	33	38	21

**SJF:**

SJF	Érkezés	CPU idő
P1	0	3
P2	1	5
P3	3	2
P4	9	5
P5	12	5

**SJF megoldás:**

SJF	Érkezés	CPU idő	Indulás	Befejezés	Várakozás
P1	0	3	0	3	0
P2	1	5	3	8	2
P3	3	2	8	10	5
P4	9	5	10	15	1
P5	12	5	15	20	3

## II. Round Robin (RR) esetén:

- Ütemezze az adott időszelét (5ms) alapján az egyes processzek (befejezési és várakozási/átlagos várakozási idő) paramétereit (ms)!
- A rendszerben lévő processzek végrehajtásának sorrendjét?
- Ábrázolja Gantt diagram segítségével az aktív/várakozó processzek futásának menetét!”

**RR:**

RR: 5ms	Érkezés	CPU idő
P1	0	3
P2	1	8
P3	3	2
P4	9	20
P5	12	5

**RR megoldás:**

RR: 5ms	Érkezés	CPU idő	Indulás	Befejezés	Várakozás	Várakozó processz
P1	0	3	0	3	0	P2,P3
P2	1	8	3	8	2	P2,P4
P3	3	2	8	10	5	P2,P4
P4	9	20	13	18	4	P4,P5
P5	12	5	18	23	6	P4

