

Miskolci Egyetem
Gépészmérnöki- és Informatikai Kar (GÉIK)
Gazdaságinformatikus alapképzés (G 2BGI)
2020/2021 II. (tavaszi) félév

Operációs rendszerek BSc (GEIAL302A-B)

tantárgy

2021 tavasz féléves feladat



MISKOLCI
E G Y E T E M
UNIVERSITY OF MISKOLC

Készítette:

Név: Szeli Márk

NEPTUN kód: B8VNQ7

Miskolc
2021.

Feladat leírása

3. Írjon C nyelvű programot, ami:

- létrehoz két gyermekprocesszt,
- ezek a gyermekprocesszek létrehoznak 3-3 további gyermeket,
- ezek az unokák várnak néhány másodpercet és szűnjenek meg,
- a szülők várják meg a gyermekek befejeződését és csak utána szűnjenek meg.

A feladat elkészítésének lépései

Első lépésben a szükséges header állományokat deklarálom, ugyanis a *sys/wait.h* állomány nélkül nem működne programom.

```
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <stdlib.h>
4  #include <sys/wait.h> // Szükséges, hiányos header állomány deklarálása.
```

Következő lépésként a feladat leírását kiírom megjegyzésként.

```
6  /*
7  Feladat leírása:
8  -----
9      3. Írjon C nyelvű programot, ami:
10         - létrehoz két gyermekprocesszt,
11         - ezek a gyermekprocesszek létrehoznak 3-3 további gyermeket,
12         - ezek az unokák várnak néhány másodpercet és szűnjenek meg,
13         - a szülők várják meg a gyermekek befejeződését és csak utána szűnjenek meg.
14  */
```

Ezután a *main* főprogramban deklarálom két különböző tömböt. Első tömb neve: *gyermek_pid*, típusa *int*, mérete 2, amely a gyermekek PID (Process ID) értékét tárolja. Második tömb neve: *unoka_pid*, típusa szintén *int*, mérete 6, amely az unokák PID (Process ID) értékét tárolja. A Process ID értéke az adott processz egyéni azonosítóját jelenti.

```
16  int main()
17  {
18      int gyermek_pid[2]; // Gyermekprocesszek PID értékeinek tárolására szolgáló tömb.
19      int unoka_pid[6]; // Unokaprocesszek PID értékeinek tárolására szolgáló tömb.
```

A tömbök deklarálása után kiíratom a képernyőre a szülőprocessz létezését.

```
21      printf("Szülőprocessz vagyok!\n"); // Szülőprocessz kiírása.
```

Ezután indítok egy számláló (*for*) ciklust, amely a szülőprocessz gyermekeit (két gyermekprocessz) hozza létre. Mivel két gyermeket kell létrehozni, az *i* változó 0-tól indul és <2-ig megy, így két gyermekprocessz kerül létrehozásra.

```
23      for(int i = 0; i < 2; i++) // 2 gyermekprocessz létrehozását szolgáló számláló (for) ciklus.
24      {
```

Jelen elágazásban, ha a feltétel teljesül (azaz a *fork()* függvény visszatérési értéke 0), akkor lefut az *if* ág, azaz létrehozásra kerülnek a gyermekprocesszek.

```
25     if(fork() == 0) // A gyermekprocessz létrehozása. Ha a fork() rendszerhívás visszatérési értéke 0, akkor lefut az elágazás.
26     {
```

Az elágazáson belül először beállítom, hogy a tömb (*gyermek_pid*) *i*-edik eleme legyen a létrehozott gyermekprocessz PID-je.

```
27         gyermek_pid[i] = getpid(); // A tömb i-edik elemének beállítása.
```

A beállítás után kiírtam a gyermekprocessz létezését, majd annak PID értékét (*getpid()*), valamint szülőprocesszének PID értékét (*getppid()*).

```
29         printf("Gyermekprocessz vagyok! PID értékem: %d, Szülőm PID értéke: %d\n", getpid(), getppid()); // A gyermekprocessz, annak PID-je, valamint a szülőjének PID-jének kiírása.
```

A kiírás után jöhet a gyermekprocesszek gyermekprocesszének, azaz az unokaprocesszek létrehozása. A létrehozás hasonló, mint a gyermekprocessz létrehozása. Indítok egy újabb számláló (*for*) ciklust, amely a gyermekprocessz gyermekeit (három unokaprocesszt) hozza létre. Abban különbözik az előzőtől, hogy az *i* értéke <3-ig megy, ezáltal három-három gyermekprocesszt (összesen hat unokaprocesszt) hoz létre.

```
31     for(int i = 0; i < 3; i++) // 3-3 unokaprocesszek létrehozására szolgáló számláló (for) ciklus.
```

Jelen elágazásban, ha a feltétel teljesül (azaz a *fork()* függvény visszatérési értéke 0), akkor lefut az *if* ág, azaz létrehozásra kerülnek az unokaprocesszek.

```
33     if(fork() == 0) // Az unokaprocesszek létrehozása. Ha a fork() rendszerhívás visszatérési értéke 0, akkor lefut az elágazás.
34     {
```

Az elágazáson belül először beállítom, hogy a tömb (*unoka_pid*) *i*-edik eleme legyen a létrehozott unokaprocessz PID-je.

```
35         unoka_pid[i] = getpid();
```

A beállítás után kiírtam az unokaprocesszek létezését, majd azok PID értékét (*getpid()*), valamint szülőprocesszüknek (azaz a gyermekprocesszek) PID értékét (*getppid()*).

```
36         printf("Unokaprocessz vagyok! Unokaprocessz PID értékem: %d, Szülőm PID értéke: %d\n", getpid(), getppid()); // Az unokaprocessz, annak PID-je, valamint a szülőjének PID-jének kiírása.
```

Az unokaprocesszek 10 másodpercet várnak.

```
37         sleep(10); // Az unokaprocesszek 10 másodperc után megszűnnek.
```

A megszűnés eredményét kiíratom, valamint azt is, hogy mely PID értékű unokaprocessz szűnt meg.

```
38         printf("Megszűnt unokaprocessz! A megszűnt unokaprocessz PID értéke: %d\n", getpid()); // Az unokaprocessz megszűnésének kiírása, valamint annak PID-je.
```

Majd *exit*ál, befejeződik a processzek élettartama.

```
39         exit (0);
```

Az elágazáson kívül a *wait(NULL)* paranccsal megvárjuk, hogy az unokaprocesszek megszűnjenek.

```
43     wait(NULL); // Unokaprocesszekre való várakozás.
```

A megszűnés eredményét kiíratom, valamint azt is, hogy mely PID értékű unokaprocessz szűnt meg.

```
44 printf("Megszűnt gyermekprocessz! A megszűnt gyermekprocessz PID értéke: %d\n", getpid()); // A gyermekprocessz megszűnésének kiírása, valamint annak PID-je.
```

Majd exitál, befejeződik a processzek élettartama.

```
45 exit (0);
```

Az elágazáson és a számláló cikluson kívül a *wait(NULL)* paranccsal megvárjuk, hogy a gyermekprocesszek megszűnjenek.

```
51 wait(NULL); // Gyermekprocesszekre való várakozás.
```

A megszűnés eredményét kiíratom, valamint azt is, hogy mely PID értékű unokaprocessz szűnt meg.

```
52 printf("Megszűnt szülőprocessz!\n"); // Szülőprocessz megszűnésének kiírása.
```

Végül megszűnik a szülőprocessz is.

```
53 return 0;
```

A program teljes képe:

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <stdlib.h>
4 #include <sys/wait.h> // Szükséges, hiányos header állomány deklarációja.
5
6 /*
7  feladat leírása:
8  -----
9  3. Írjon C nyelvű programot, ami:
10     - létrehoz két gyermekprocesszt,
11     - ezek a gyermekprocesszek létrehozhatnak 3-3 további gyermeket,
12     - ezek az unokák várakoznak néhány másodpercet és szűnnek meg,
13     - a szülők várják meg a gyermekek befejeződését és csak utána szűnnek meg.
14 */
15
16 int main()
17 {
18     int gyermek_pid[2]; // Gyermekprocesszek PID értékeinek tárolására szolgáló tömb.
19     int unoka_pid[6]; // Unokaprocesszek PID értékeinek tárolására szolgáló tömb.
20
21     printf("Szülőprocessz vagyok!\n"); // Szülőprocessz kiírása.
22
23     for(int i = 0; i < 2; i++) // 2 gyermekprocessz létrehozását szolgáló számláló (for) ciklus.
24     {
25         if(fork() == 0) // A gyermekprocessz létrehozása. Ha a fork() rendszerhívás visszatérési értéke 0, akkor lefut az elágazás.
26         {
27             gyermek_pid[i] = getpid(); // A tömb i-edik elemének beállítása.
28
29             printf("Gyermekprocessz vagyok! PID értéke: %d, Szülőm PID értéke: %d\n", getpid(), getppid()); // A gyermekprocessz, annak PID-je, valamint a szülőjének PID-jének kiírása.
30
31             for(int j = 0; j < 3; j++) // 3-3 unokaprocesszek létrehozására szolgáló számláló (for) ciklus.
32             {
33                 if(fork() == 0) // Az unokaprocesszek létrehozása. Ha a fork() rendszerhívás visszatérési értéke 0, akkor lefut az elágazás.
34                 {
35                     unoka_pid[i*3+j] = getpid();
36                     printf("Unokaprocessz vagyok! Unokaprocessz PID értéke: %d, Szülőm PID értéke: %d\n", getpid(), getppid()); // Az unokaprocessz, annak PID-je, valamint a szülőjének PID-jének kiírása.
37                     sleep(10); // Az unokaprocesszek 10 másodperc után megszűnnek.
38                     printf("Megszűnt unokaprocessz! A megszűnt unokaprocessz PID értéke: %d\n", getpid()); // Az unokaprocessz megszűnésének kiírása, valamint annak PID-je.
39                     exit (0);
40                 }
41             }
42
43             wait(NULL); // Unokaprocesszekre való várakozás.
44             printf("Megszűnt gyermekprocessz! A megszűnt gyermekprocessz PID értéke: %d\n", getpid()); // A gyermekprocessz megszűnésének kiírása, valamint annak PID-je.
45             exit (0);
46         }
47     }
48
49     wait(NULL); // Gyermekprocesszekre való várakozás.
50     printf("Megszűnt szülőprocessz!\n"); // Szülőprocessz megszűnésének kiírása.
51     return 0;
52 }
```

A futtatás eredménye

```
~$ ./beadando
Szülőprocessz vagyok!
Gyermekprocessz vagyok! PID értékem: 1315, Szülőm PID értéke: 1314
Unokaprocessz vagyok! Unokaprocessz PID értékem: 1317, Szülőm PID értéke: 1315
Unokaprocessz vagyok! Unokaprocessz PID értékem: 1318, Szülőm PID értéke: 1315
Unokaprocessz vagyok! Unokaprocessz PID értékem: 1319, Szülőm PID értéke: 1315
Gyermekprocessz vagyok! PID értékem: 1316, Szülőm PID értéke: 1314
Unokaprocessz vagyok! Unokaprocessz PID értékem: 1320, Szülőm PID értéke: 1316
Unokaprocessz vagyok! Unokaprocessz PID értékem: 1321, Szülőm PID értéke: 1316
Unokaprocessz vagyok! Unokaprocessz PID értékem: 1322, Szülőm PID értéke: 1316
Megszűnt unokaprocessz! A megszűnt unokaprocessz PID értéke: 1317
Megszűnt unokaprocessz! A megszűnt unokaprocessz PID értéke: 1318
Megszűnt unokaprocessz! A megszűnt unokaprocessz PID értéke: 1319
Megszűnt unokaprocessz! A megszűnt unokaprocessz PID értéke: 1320
Megszűnt unokaprocessz! A megszűnt unokaprocessz PID értéke: 1321
Megszűnt unokaprocessz! A megszűnt unokaprocessz PID értéke: 1322
Megszűnt gyermekprocessz! A megszűnt gyermekprocessz PID értéke: 1316
Megszűnt gyermekprocessz! A megszűnt gyermekprocessz PID értéke: 1315
Megszűnt szülőprocessz!
~$ █
```