Jegyzőkönyv

Operációs rendszerek BSc

Készítette:

Krakkai Renátó Tibor Bsc

Mérnökinformatikus

Neptunkód: PIP7QV

Miskolc, 2021

A feladat leírása:

3. Írjon C nyelvű programot, ami:
- létrehoz két gyermekprocesszt
- ezek a gyermekprocesszek létrehoznak 3-3 további gyereket
- ezek az unokák várakozzanak néhány másodpercet és szűnjenek meg
- a szülők várják meg a gyerekek befejeződését és csak utána szűnjenek meg

A feladat elkészítésének lépései:

A program futtatásához C környezetben egy main() függvényre van szükség.
A main() függvény törzsében végezzük a feladatokat, más általunk írt függvényre nincs szükségem a feladat megoldásához.

A program futásakor először kiírjuk a fő process id-ját a képernyőre a getpid() függvény segítségével, így ismert a fő process ID-ja.

Ezután létrehozunk két gyerek processt a fork() függvény segítségével, ez pid_t típusú változó, ami a process ID-ját tartalmazza. Ha ez nulla, akkor a process létrehozása sikertelen volt. A forkolt processek nevei: firstChild és secondChild lesznek a program futtatása során.

Egy if elágazással ellenőrizzük, hogy a a fork sikeres volt-e. Ilyenkor azt is ellenőrizni kell, hogy a fő process futtatja-e a kódot, vagy valamelyik gyermek process. A gyerekek az if elágazás true ágába mennek, a szülő pedig az else ágat hajtja végre.

Az if elágazás true ágában kiírjuk a gyerek processek process id-ját, valamint a szülőét is, önellenőrzés céljából. A szülök process ID-ját a getppid() függvénnyel tudjuk lekérni.

Létrehozunk egy status nevű, int típusú változót, amiben majd a processek kilépési státuszát tudjuk tárolni. Ezt a státuszt később nem használjuk.

Lértehozunk egy children nevű pid_t típusú 3 elemű tömböt, melyben a további gyermek processeket tudjuk tárolni.

Egy for függvény segítségével a children tömböt feltöltjjük elemekkel (children[i] = fork();)

A for függvény további részében az unokák által futtatott kódot írjuk le.

A for függvény után a waitpid() függvény segítségével megvárjuk, amíg az unoka processek befejezik a futásukat. Miután minden unoka process befejezte a futását, az exit() függvénnyel kilépünk a gyerek processből.

Az unokák által futtatott kód (20-25. sor) a következőt csinálja:

Kiírja az unokat process id-ját, valamint a szülő pricess id-ját a getpid() és getppid() függvények segítségével.

Vár 3 másodpercet (sleep(3))

Kilép 0-s visszatérési értékkel (exit(0))

A fő program további részében (33-49. sor) a következők történnek:

Létrehoutunk egy-egy változót az első és második gyerek kilépési státuszának a tárolására

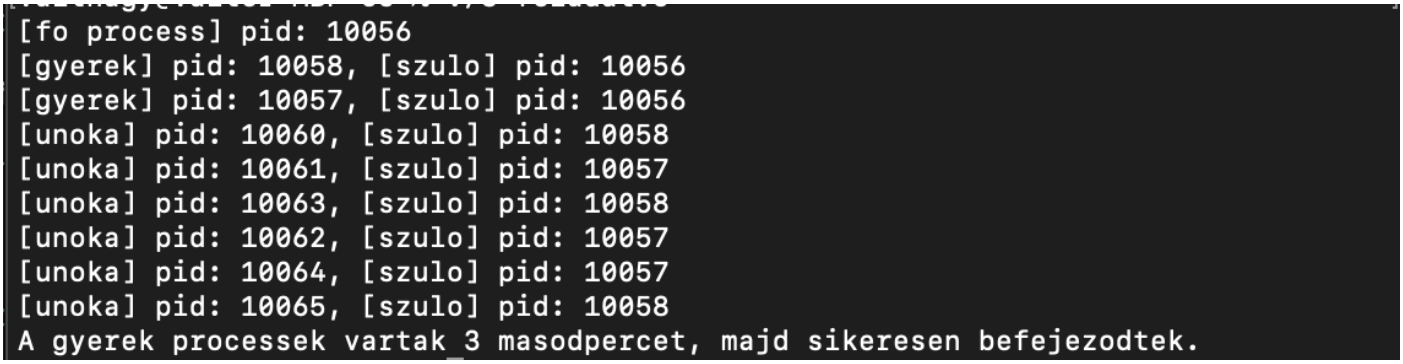
A waitpid() függvénnyel megvárljuk az első és a második gyerek futásának befejeződését, és ezeket elmentjük az imént a kilépési stáruszok mentésére létrehozott változókba.

Egy if elágazás segítségével ellenőrizzük, hogy mindkét gyermek futása sikeres volt-e, nincs a kilépési stásuszok között 0-tól eltérő érték.

Ha mindkét gyerek sikeresen befejezte a futását, akkor jelezzük a sikeres futást, ellenkező esetben is értesítjük a felhasználót a hibáról.

```c
#include <stdlib.h>
#include <stdio.h>
#include <sys/wait.h>
#include <unistd.h>

int main() {
    printf("[fo process] pid: %d\n", getpid());

    pid_t firstChild = fork();
    pid_t secondChild = fork();

    if((firstChild == 0 || secondChild == 0) && (firstChild != 0 || secondChild != 0) )  // fork succeeded
    {
        printf("[gyerek] pid: %d, [szulo] pid: %d\n", getpid(), getppid());
        int status;
        pid_t children[3];
        for (int i = 0; i < 3; i++ ) {
            children[i] = fork();

            if (!(children[0] != 0 && children[1] != 0 && children[2] != 0)) // ha nem fo process
            {
                printf("[unoka] pid: %d, [szulo] pid: %d\n", getpid(), getppid());
                sleep(3);
                exit(0);
            }
        }
        waitpid(children[0], &status, 0);
        waitpid(children[1], &status, 0);
        waitpid(children[2], &status, 0);

        exit(0);
    }
    else if (firstChild != 0 && secondChild != 0)  // Main (parent) process after fork succeeds
    {
        int firstChildReturnStatus;
        int secondChildReturnStatus;

        waitpid(firstChild, &firstChildReturnStatus, 0);  // Parent process waits here for child to terminate.
        waitpid(secondChild, &secondChildReturnStatus, 0);  // Parent process waits here for child to terminate.

        if (firstChildReturnStatus == 0 && secondChildReturnStatus == 0)  // Verify child process terminated without error.
        {
           printf("A gyerek processek vartak 3 masodpercet, majd sikeresen befejezodtek.\n");
        }
        else
        {
            printf("A gyerekek hibaval fejeleződtek be.\n");
        }
    }
}
```

A futtatás eredménye:



A futás eredményéből látszik, hogy a fő process (pid: 10056) létrehozott két gyermek processt (10058 és 10057), melyek létrehoztak 3-3 unokát (10060-10063-10065 ; 10061-10062-10064). Itt a végrehajtás közben érzékelt idő körülbelül megfelelt 3 másodpercnek, amíg a gyerek és unoka processek be nem fejezték a futásukat, majd megtörtént az utolsó sor kiírása is, ami a helyes futásra enged következtetni.