

JEGYZŐKÖNYV

Operációs rendszerek BSc

2022. tavasz féléves feladat

Készítette: **Danyi Kristóf Milán**

Neptunkód: **GQOKMW**

A feladat leírása:

4. Adott az alábbi terhelés esetén a rendszer. Határozza meg az *indulás*, *befejezés*, *várakozás/átlagos várakozás és körülfordulás/átlagos körülfordulás*, *válasz/átlagos válaszidő* és a *CPU kihasználtság* értékeket az SJF ütemezési algoritmusok mellett! (cs: 0,1ms; sch: 0,1ms)

	P1	P2	P3	P4
Érkezés	0	8	12	20
CPU idő	15	7	26	10
Indulás				
Befejezés				
Várakozás				

4. Írjon C nyelvű programot, ami
letréhoz két csövet (ket file deszkriptor part)
elforkol
a szülő elküldi a saját pid-jét a gyermeknek az egyik csőon
a gyermek kiírja a képernyőre és visszküldi egy az övet a másik csőon
megszűnnek a processzek (a szülő megvárja a gyereket)

A feladat elkészítésének lépései: Az excelben egyszerű számolásokat végeztem, a C programban pedig úgy érzem, hogy a forráskódot megfelelő mértékben kommenteltem, így felhasználom a feladat lépéseinek magyarázataként

```
*beadando.c (~/Downloads)
File Edit View Search Tools Documents Help
[Icons]

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#define SIZE 256

int main()
{
    int fd1[2]; //Létrehozzuk az első csövet
    int fd2[2]; //Létrehozzuk a második csövet
    int parentmsg; //Ide mentjük a processz ID-t
    int childmsg;

    if(pipe(fd1) < 0){ // Megpipeoljuk az fd1 és fd2-t
        exit(1);
    }
    if(pipe(fd2) < 0){
        exit(1);
    }

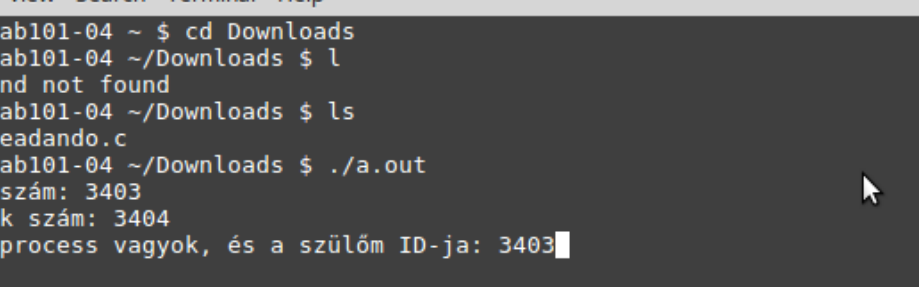
    pid_t pid = fork(); //létrehozzuk a child process

    if(pid != 0){
        //szülő részleg
        pid_t Ppid = getpid(); //Eltároljuk a szülő Pid-jét
        printf("A szülő szám: %d\n", Ppid);
        write(fd1[1], &Ppid, SIZE); //Beleírjuk az első csőbe a szülő pid-et
        wait(NULL); //Megvárjuk még a gyermek process megcsinálása a dolgait
        read(fd2[0], &childmsg, SIZE); //Kiolvassuk a második csőből a gyermek pid-et
        kill(childmsg, SIGTERM); //Megöljük a gyermeket
        kill(Ppid, SIGTERM); // Utána Megöljük a szülőt
    }

    else if (pid == 0){
        //gyermek részleg
        pid_t Cpid = getpid(); //Eltároljuk a gyermek pid-et
        printf("A gyermek szám: %d\n", Cpid);
        read(fd1[0], &parentmsg, SIZE); //Kiolvassuk a szülő pid-et az első csőből
        write(fd2[1], Cpid, SIZE); //Beleírjuk a második csőbe a gyermek pid-et
        printf("A Child process vagyok, és a szülőm ID-ja: %d", parentmsg); //Kiíratjuk a gyermekkel a szülő pidet
    }

    return 0;
}
```

A futtatás eredménye:



The screenshot shows a macOS Terminal window with the title "Terminal". The menu bar includes "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal content shows a user named "danyi3" at host "lab101-04" in the directory "~". The user enters the command "cd Downloads", then "l" (which results in "l: command not found"), then "ls" (showing "a.out" and "beadando.c"), and finally runs the program with "./a.out". The program outputs three lines: "A szülő szám: 3403", "A gyermek szám: 3404", and "A Child process vagyok, és a szülőm ID-ja: 3403". A mouse cursor is visible on the right side of the terminal window.

```
danyi3@lab101-04 ~ $ cd Downloads
danyi3@lab101-04 ~/Downloads $ l
l: command not found
danyi3@lab101-04 ~/Downloads $ ls
a.out  beadando.c
danyi3@lab101-04 ~/Downloads $ ./a.out
A szülő szám: 3403
A gyermek szám: 3404
A Child process vagyok, és a szülőm ID-ja: 3403
```

[illegible]