# JEGYZŐKÖNYV

## Operációs rendszerek BSc

## 2022. tavasz féléves feladat

Készítette: **Szkárosi Szilárd**

Neptunkód: **DLWGQZ**

**1. feladat:**

# IPC mechanizmus

**A feladat leírása:**

4.feladat: Írjon C nyelvű programot, ami létrehoz két csővezetéket (két file deszkriptor part) elforkol. A szülő elküldi a saját pidjét a gyermeknek az egyik csövön. A gyermek kiirja a képernyőre és visszaküldi egy az övét a másik csövön. Megszűnnek a processzek (a szülő megvárja a gyereket).

**A feladat elkészítésének lépései:**

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

int main()
{
    //Ket darab valozot hoztam letre, hogy krealni tudjam a ket csovezeteket
    int pipefd[2];
    int pipefdMasik[2];

    pid_t cpid;           //A gyerek pidje lesz benne
    int buf;              //Letrehoztam egy buffert
    int bufMasik;         //Masik buffer a kuldendo szovegnek

    if (pipe(pipefd) == -1)
    {
        perror("pipe");    //Csovezeteket krealunk, amiben a ket fd lesz eltarolva
        exit(-1);
    }

    if (pipe(pipefdMasik) == -1)
    {
        perror("pipe");   //Csovezeteket krealunk, amiben a ket fd lesz eltarolva
        exit(-1);
    }

    cpid = fork();
    if (cpid == -1)
    {
        perror("Elforkol");
        exit(-1);
    }
```

A feladat leírása alapján két darab csővezetéket kell elforkolni. Ennek megvalósítására két darab változót deklaráltam, amelynek segítségével kreálni tudom a csővezetékeket.

```c
if (cpid == 0)      //A cpid gyerek lesz
{
    printf("%d: gyerek vagyok\n",getpid()); //A gyerek csak olvasni fog
    while(read(pipefd[0], &buf, 1) > 0)
        printf("%d",buf);

    //Itt becsukom az olvaso veget
    int pid = getpid();
    write(pipefdMasik[1], &pid, sizeof(int));

    exit(0); // kilepek

}
```

Ha a cpid értéke 0, akkor létrehozzuk a gyerek pid-jét. Azután a read() rendszerhívás segítségével beolvassuk a csővezetéken keresztül a gyerek pid-jét, majd egy másik csővezetéken a write() rendszerhívással kiiratom a pid-et és végül kilépek.

```c
else                //Ilyenkor a cpid szulo lesz
{
    printf("%d: szulo vagyok\n",getpid());
    //Itt az olvaso veget fogjuk bezarni, mert irni fogunk

    int pid = getpid();
    write(pipefd[1], &pid, sizeof(int)); // Anyagot ontunk a csobe

    //Addig olvasunk a cso kimeneterol, amig meg lehet (1 byte-onkent)
    //Lecsukom a csatorna fedelet, a masik oldalon EOF lesz
    while (read(pipefdMasik[0], &bufMasik, 1) > 0)
        printf("%d",bufMasik);

    wait(NULL);  //Gyerekre varunk

    exit(0);
}
```

Ellenkező esetben, ha a cpid értéke nem egyenlő 0-val, akkor a szülő pid-jét hozzuk létre. A write() rendszerhívással beleírjuk a csőbe a pid értékét, majd a másik csövön pedig a read() rendszerhívással beolvassuk a pid-et. A wait() rendszerhívás segítségével megvárjuk a gyereket.

**A futtatás eredménye:**

```
szkarosi1@szkarosi1-VirtualBox:~/OS_GYAK/OS_beadando$ ./4_f.out
3292: szulo vagyok
3293: gyerek vagyok
```

**2. feladat:**

# OS algoritmusok

**A feladat leírása:**

35.feladat: 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 FCFS ütemezési algoritmusok mellett! (cs: 0,1ms; sch: 0,1ms)

**A feladat elkészítésének lépései:**

| FCFS Ütemezés | | | | | |
|---|---|---|---|---|---|
| | P1 | P2 | P3 | P4 | P5 |
| Érkezés | 1 | 4 | 4 | 7 | 8 |
| CPU idő | 4 | 11 | 4 | 7 | 4 |
| Indulás | 1 | 5 | 16 | 20 | 27 |
| Befejezés | 5 | 16 | 20 | 27 | 31 |
| Várakozás | 1 | 5 | 16 | 20 | 27 |
| Körülfordulási idő | 4 | 12 | 16 | 20 | 23 |

| | |
|---|---|
| Körülfordulási idők átlaga | 15 |
| Válaszidők átlaga | 93 |
| CPU kihasználtság | 129 |
| Várakozási idők átlaga | 13,8 |

**FCFS Gantt-diagram:**