

JEGYZŐKÖNYV

Operációs rendszerek BSc

2022. tavasz féléves feladat

Készítette: Tucsa Eszter Boglárka

Neptunkód: G2QWPO

1. Feladat Leírása:

4. Írjon C nyelvű programot, ami:

- Létrehoz két csővezetékét (két file deszkriptor párt), elforkol.
- A szülő elküldi a saját pidjét a gyermeknek az egyik csövön.
- A gyermek kiírja a képernyőre és visszaküldi, az övét a másik csövön.
- Megszűnnek a processek (a szülő megvárja a gyereket).

```
4. Irjon C nyelvű programot, ami
  létrehoz két csővezetékét (két file deszkriptor part)
  elforkol
  a szülő elküldi a saját pidjét a gyermeknek az egyik csövön
  a gyermek kiírja a képernyőre és visszaküldi egy az övét a másik csövön
  megszűnnek a processzek (a szülő megvárja a gyereket)
```

Elkészítés lépései:

- Feladatértelmezés, fordítás magyarról-magyarra, majd magyarról körül-belüli kódra.
- Kód fejben megtervezése, nagyvonalakban, majd CodeBlocks megnyitása.
 - Megfelelő header állományok includálása, hogy a megfelelő metódusokat /műveleteket lehessen használni (pl.: getpid(), vagy a fork() az unistd.h -ban.)

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

- Feladat elemi részekre bontása, először is:
 - 2 csővezeték kell, egy a szülőnek és egy a gyerek processznek.
 - Létrehozom a csővezetéseket és kezelem a sikertelen létrehozás kivételt:

```
int main()
{
    // 2 csővezeték:
    // - parent_fds: Amit a szülő használ hogy a gyereknek írjon
    // - child_fds: Amit a gyerek használ hogy a szülőnek írjon
    int parent_fds[2], child_fds[2];

    // olvasas:[0] - iras:[1]
    if (pipe(parent_fds) != 0 || pipe(child_fds) != 0)
    {
        printf("Csővek létrehozása sikertelen!\n");
        return EXIT_FAILURE;
    }
}
```

- Ezt követően elforkolom a gyermek processt, itt is kezelem a sikertelen létrehozás kivételt:

```
// Gyerek processz forkolas
int child = fork();

if (child < 0)
{
    printf("Sikertelen forkolas!\n");
    return EXIT_FAILURE;
}
```

- Ha a fork sikeres volt, lezárom a csővezetékek megfelelő „végeit”, majd kiolvastatom a gyermekkel a parent pid-jét, és kiíratom a képernyőre az eredményt, majd lezárom a szülő vezetéket olvasásra is.

```
else if (child == 0)
{
    close(child_fds[0]);
    close(parent_fds[1]);

    // Olvasas szulo csővezetekbol
    int parenttol;
    read(parent_fds[0], &parenttol, sizeof(parenttol));
    printf("GYERÉK - PID = %d: Beolvastam a szülőtol: %d\n", (int)getpid(), parenttol);

    close(parent_fds[0]);
}
```

- Lekérdezem a gyerek pidjét és beíratom a saját csővezetékébe, hogy a szülő kiolvashassa, majd a gyerek vezetékének az író részét lezárom:

```
// PID irasa a gyerek vezetékebe hogy a szulo kiolvassa
int pid2=getpid();
write(child_fds[1], &pid2, sizeof(pid2));
close(child_fds[1]);

printf("GYEREK - PID = %d: Rairtam a PID-em a csovezetekre!\n\n", pid2);
```

- Egyéb esetben (ha a fork() pozitív értéket adott vissza) a szülő processhez ugrik, itt lezárom a csővezetéseket, lekérdezem és beíratom a szülő pidjét saját csővezetékébe, hogy azt majd a gyerek process ki tudja olvasni onnan, majd lezárom a csővezetésekre írást:

```
else
{
    // Lezaram a csovezetekeket
    close(parent_fds[0]);
    close(child_fds[1]);

    // PID irasa a szulo vezetekbe hogy a gyerek kiolvassa
    int pid=getpid();
    printf("SZULO - PID = %d: Rairtam a PID-em a csovezetekre!\n\n", pid);

    write(parent_fds[1], &pid, sizeof(pid));
    close(parent_fds[1]);
}
```

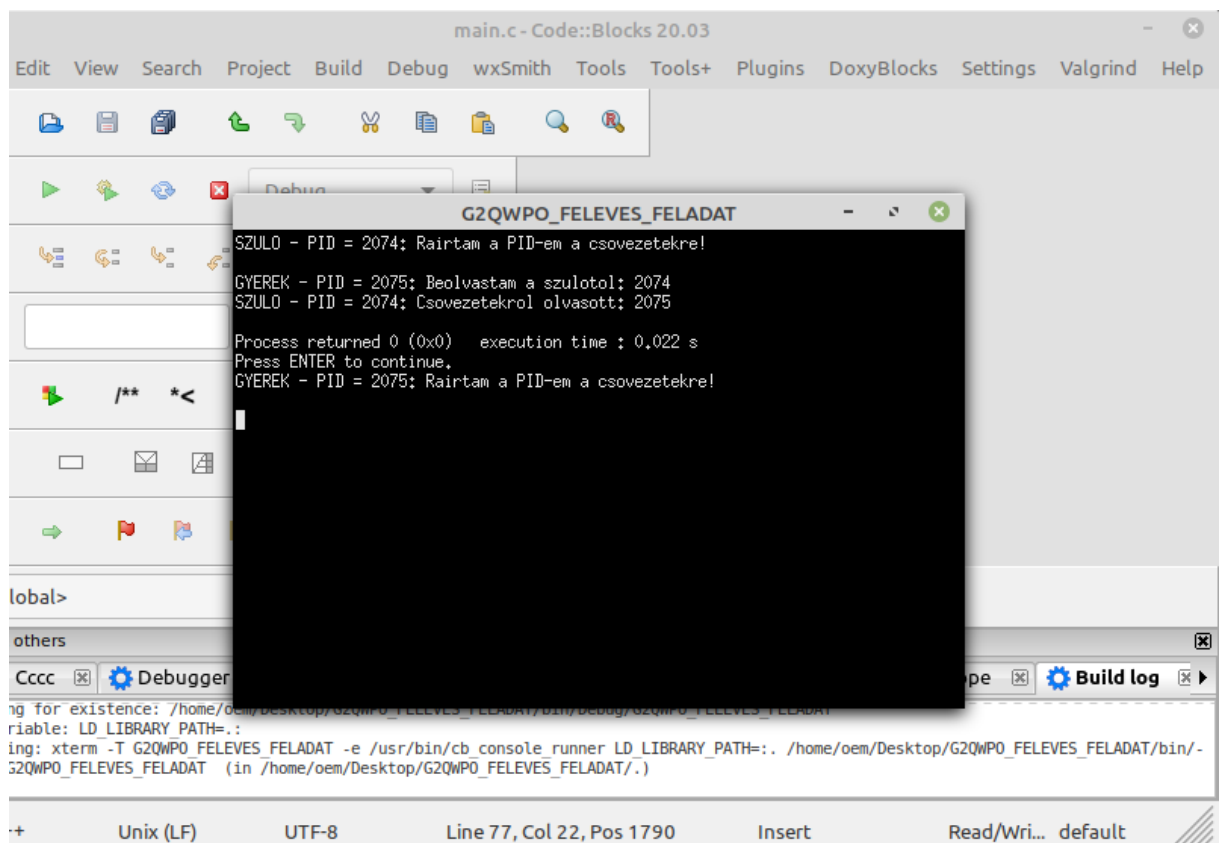
- Ekkor fog majd a gyermek process színre lépni a 0 értékkel, és végrehajtja a neki kijelölt feladatokat. Ezt követően az irányítás ismét a szülő processhez kerül, ami most kiolvassa a gyerek csővezetékéből a gyerek pidjét, majd pedig kiíratom a képernyőre:

```
// Olvasas gyerek csovezetekbol
int chldtol;
read(child_fds[0], &chldtol, sizeof(chldtol));

close(child_fds[0]);
printf("SZULO - PID = %d: Csovezetekrol olvasott: %d\n", (int) getpid(), chldtol);
}
return 0;
}
```

- Lefuttatom, örülök, hogy működik, szépen lejegyzem a jegyzőkönyvbe, amit lekell, és beillesztem a futtatási eredményt.

A futtatás eredménye:



```
main.c - Code::Blocks 20.03
Edit View Search Project Build Debug wxSmith Tools Tools+ Plugins DoxyBlocks Settings Valgrind Help

G2QWPO_FELEVES_FELADAT
SZULO - PID = 2074: Rairtam a PID-em a csovezetekre!
GYEREK - PID = 2075: Beolvastam a szulotol: 2074
SZULO - PID = 2074: Csovezetekrol olvasott: 2075
Process returned 0 (0x0) execution time : 0.022 s
Press ENTER to continue.
GYEREK - PID = 2075: Rairtam a PID-em a csovezetekre!
```

2. Feladat:

9. Adott az alábbi terhelés esetén a rendszer. határozza meg az *indulás*, *befejezés*, *várakozás*, *átlag várakozás*, *körülfordulás*, *átlagos körülfordulás*, illetve *válaszidő* és *átlag válaszidő*, valamint a *CPU kihasználtság* értékeit az RR:5ms ütemezési algoritmusok mellett! (cs: 0,1ms, sch: 0,1ms) (kiinduló táblázat a képen) Ábrázolja Gantt diagram segítségével az *aktív/várakozó processzek* futásának menetét! Magyarázza a kapott eredményt!

9. Adott az alábbi terhelés esetén a rendszer. Határozza meg az *indulás*, *befejezés*, *várakozás/átl várakozás* és *körülfordulás/átlagos körülfordulás*, *válasz/átl. válaszidő* és a *CPU kihasználtság* értékeket az RR:5 ms 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				

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

Magyarázza a kapott eredményeket!

Megoldás lépései:

- Először, is agyi fogaskerek beindítása:
 - RR=Adott process csak a megadott időhosszúságig futhat. Ez itt 5ms, tehát egyszerre egy process maximum 5 ms-ig futhat, ennek leteltével a process a befejezés idejében újra érkezik, a CPU időigénye viszont az 5ms-sel csökkenni fog (kivéve, ha az időigény kevesebb volt a max kiszabottnál, akkor már

nem fog ismét érkezni és nem marad semmi igénye, ő kész van, befejezte végleg a futást.). Utána megnézzük, hogy a befejezési ideig érkezett-e másik process, ha igen, akkor az fog maximum a max kiszabott ideig futni. És ez így megy tovább.

- Indulási idő: 0, ha az első futó process, ellenkező esetben mindig az előző process befejezési ideje.
- Várakozás: Az érkezés és az indulás hányadosa, azt mutatja meg, mennyit kellett várnia a processnek a futásáig. Az átlag várakozás ezeknek az átlaga.
- Körülfordulás: Mennyi időbe telik, amég a process teljesen lefut. A várakozási idő és a process végrehajtásához szükséges idő összege. Ezek átlaga az átlag körülfordulás.
- Válaszidő: Egy process érkezéstől számított első reakciójáig eltelt idő. (Tulajdonképpen RR esetén mindenhol az első várakozási idők, FCFS SJF esetén megegyezik a várakozási időkkel.) Az átlag válaszidő ezeknek az átlaga.
- Schedule (sch): Processek (újra) rendezése. Ez történik minden olyan esetben, amikor az összes bent lévő process végigfutott (legalább 1x a max időkeretig). Jelen esetben ez a „rendező folyamat” 0,1ms-ig tart.
- Context Switch (cs): Process váltás esetén. Pl.: P1 befejezi futását, ekkor a futás joga egy tfh. P2 processhez kerül, akkor egy context switch van köztük. Jelen esetben ez is 0,1ms.

- CPU kihasználtság: Az összes [CPU idő+összes cs+összes sch] (tehát a teljes-teljes CPU idő)-[összes cs+összes sch] (, mivel azért mégse process fut ezidő alatt úgyhogy mégiscsak olyan kis haszontalan idő) / a teljes-teljes CPU idő (mert azért mégiscsak fontos a schedule meg a context switch is). Ez *100 és megvan százalékosan.
- Következő lépés: szépen átláthatóan kézzel megoldani (én papíron szeretem, Excelben hamarabb elnézem a sorokat-oszlopokat).
- Ezt követően (ha pl.: esetemben papíron csináltam) bemásolni Excel-be. Szépen formázni, hogy átlátható legyen.
- Segéd táblázat létrehozása csak és kizárólag azért, hogy egyáltalán lehessen róla Gantt diagramot csinálni.
- Képernyőkép, és a jegyzőkönyvbe szépen beilleszteni.

Az eredmények:

1	Féléves Feladat									
2	RR:5ms	P1	P2	P3	P4					
3	Érkezés	0,5,10	8, 15	12, 20, 37, 47, 52, 57	20, 25					
4	CPU idő	15, 10, 5	7, 2	26, 21, 16, 11, 6, 1	10, 5					
5	Indulás	0, 5, 25	10, 30	15, 32, 42, 47, 52, 57	20, 37					
6	Befejezés	5, 10, 30	15, 32	20, 37, 47, 52, 57, 58	25, 42					
7	Várakozás	0, 0, 15	2, 15	3, 12, 5, 0, 0, 0	0, 12					
8	Körfordulási idő	30	24	46	22					
9	Válaszidő	0	2	3	0					
10										
11	CPU kihasználtság:	98%		sch:	0,1ms	3x				
12	Átlag várakozási idő:	7,7		cs:	0,1ms	8x				
13	Átlag körfordulási idő:	30,5								
14	Átlag válaszidő:	1,25								
15										
16	RR:5ms	P1	P2	P3	P4	P1-2kor	P2-2kor	P3-2kor	P4-2kor	P3-tobbi
17	Érkezés	0	8	12	20	10	15	20	25	37
18	CPU idő	15	7	26	10	5	2	21	5	16
19	Indulás	0	10	15	20	25	30	32	37	42
20	Befejezés	10	15	20	25	30	32	37	42	58
21	Várakozás	0	2	3	0	15	15	17	12	5
22	Körf: sum(vár)+sum(cpu)									
23		30	24	46	22					
24										
25										
26										
27										

RR: 5MS

■ Érkezés ■ CPU idő ■ Indulás ■ Befejezés ■ Várakozás

P3-TOBBI

P4-2KOR

P3-2KOR

P2-2KOR

P1-2KOR

P4

P3

P2

P1

<- szép diagramm segéd