

JEGYZŐKÖNYV

Operációs rendszerek BSc

2022. tavasz féléves feladat

Készítette: **Pázmán András**

Neptunkód: **H2Z4X3**

A feladat leírása:

2. 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)

	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!

A feladat elkészítésének lépései:

- 1) Kiszámítani a indulás, befejezés, várakozás, körülfordulás, válaszidőt, CPU kihasználtság

Indulás: Amikor az előző processz befejezte a futását

Befejezés: Indulás + CPU idő

Várakozás: Indulás – Érkezés

Körülfordulás: Befejezés – Érkezés

Válaszidő: Indulás – Érkezés

CPU kihasználtság: Legnagyobb(utolsó) befejezés / (utolsó befejezés + SCH * processzek száma)

- 2) Kiszámítjuk az átlagokat
- 3) Elkészítjük a a Gantt diagrammot

A futtatás eredménye:

FCFS	P1	P2	P3	P4
Érkezés	0	8	12	20
CPU idő	15	7	26	10
Indulás	0	15	22	48
Befejezés	15	22	48	58
Várakozás	0	7	10	28
Körülfordulás	15	14	36	38
Válaszidő	0	7	10	28

A feladat le írása

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

A feladat elkészítésének lépései

- 1) `fork()` rendszerhívással létrehozzuk a feladatban kért processzeket
- 2) if vizsgálatban kiírjuk az állapotot hogy melyik ágba járunk,
- 3) `sleep()`-el várakozunk, a szülők megvárják a gyerekek befejeződését majd azok is megszűnnek

*/*részletesebb dokumentáció a forráskódban*/*

A futtatás eredménye:

The screenshot shows a Sublime Text editor window titled `~/beadando/3.c - Sublime Text (UNREGISTERED)` with several tabs open. The active tab is `3.c`, which contains the following C code:

```
1 #include<stdio.h>
2 #include<sys/types.h>
3 #include<unistd.h>
4 #include<sys/wait.h>
5
6
7 int main(){
8
9     //szülők
10    int sz1, sz2;
11
12    //gyerekek
13    int gy1, gy2, gy3;
14    int gy11, gy12, gy13;
15    int sig;
16
17    //fork rendszerhívással létrehozzuk a 3. gyereket
18    sz1=fork(); // a child 0 a parent
19
20    if(sz1==0)
21    {
22        printf("$sz1 pid: %d\n", getpid());
23        //Létrejön a 3 gyermekprocessz
24        gy1=fork();
25
26        if(gy1==0)
27        {
28            printf("<in gy1 pid: %d\n", getpid());
29            sleep(3); //Varakozás
30
31            /*
32             * A sleep() rendszerhívás
33             * a sleepnek ne ma signalozzunk
34             */
35        }
36        gy2=fork();
37        if(gy2==0)
38        {
39            printf("<in gy2 pid: %d\n", getpid());
40            sleep(3);
41        }
42    }
43 }
```

The terminal window, titled `pazman@pazman-VirtualBox: ~/beadando`, shows the execution of the program:

```
pazman@pazman-VirtualBox:~/beadando$ cd beadando
pazman@pazman-VirtualBox:~/beadando$ gcc 3.c -o 3
pazman@pazman-VirtualBox:~/beadando$ ./3
varakozas.....
$sz1 pid: 21012
<in gy1 pid: 21013
<in gy2 pid: 21014
<in gy3 pid: 21015
$in sz2 pid: 21016
[In gy11 pid: 21017
[In gy12 pid: 21018
[In gy13 pid: 21019
<in gy2 pid: 21020
$in sz2 pid: 21024
$in sz2 pid: 21026
[In gy13 pid: 21027
<in gy3 pid: 21023
<in gy3 pid: 21021
[In gy13 pid: 21033
[In gy13 pid: 21028
[In gy12 pid: 21031
[In gy13 pid: 21034
$in sz2 pid: 21022
[In gy11 pid: 21030
[In gy11 pid: 21029
[In gy12 pid: 21032
[In gy11 pid: 21035
```