Operációs rendszerek

6.Gyakorlat

2025.03.26.

Készítette:

Orosz Kristóf Bsc

Szak: Programtervező Informatikus

Neptunkód: EYZWG9

Sárospatak, 2025

I. Határozza meg FCFS, SJF és RR esetén

a.) A befejezési időt?

FCFS	P1	P2	P3	P4	P5
Érkezés	0	1	3	9	12
CPU idő	3	8	2	20	5
Indulás	0	3	11	13	33
Befejezés	3	11	13	33	38

SJF				
Processz	Érkezési idő	CPU igény	Kezdési idő	Befejezési idő
F1	0	3	0	3
F2	1	5	5	10
F3	3	2	3	5
F4	9	5	10	15
F5	12	5	15	20

Round Robin					
5 ms	P1	P2	P3	P4	P5
Érkezés	0	1	3	9	12
CPU idő	3	8	2	20	5
Indulás	0	3	8	13	18
Befejezés	3	8,13	10	18,38	23

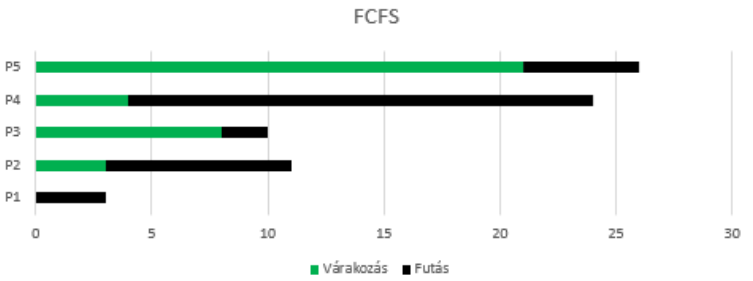
b.) A várakozási/átlagos várakozási időt?

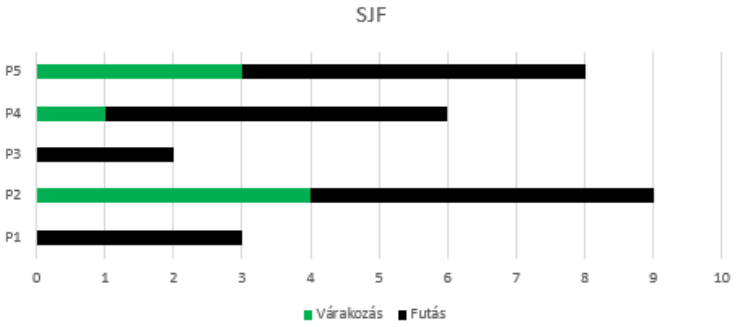
FCFS	P1	P2	P3	P4	P5
Érkezés	0	1	3	9	12
CPU idő	3	8	2	20	5
Indulás	0	3	11	13	33
Befejezés	3	11	13	33	38
Várakozás	0	2	8	4	21

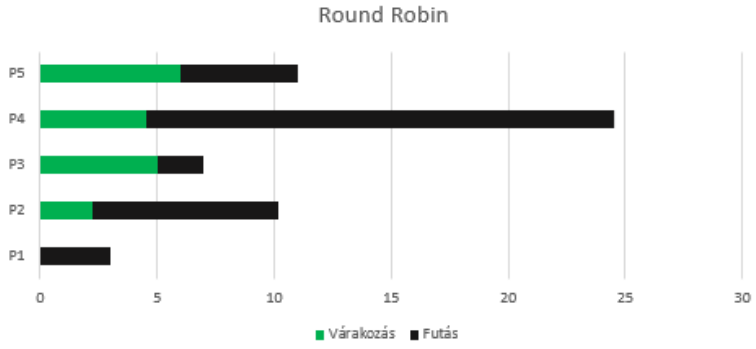
SJF					
Processz	Érkezési idő	CPU igény	Kezdési idő	Befejezési idő	Várakozási idő
F1	0	3	0	3	0
F2	1	5	5	10	4
F3	3	2	3	5	0
F4	9	5	10	15	1
F5	12	5	15	20	3

Round Robin					
5 ms	P1	P2	P3	P4	P5
Érkezés	0	1	3	9	12
CPU idő	3	8	2	20	5
Indulás	0	3	8	13	18
Befejezés	3	8,13	10	18,38	23
Várakozás	0	2,2	5	4,5	6

c.) Ábrázolja Gantt diagram segítségével az aktív/várakozó processzek futásának menetét.
Megj.: a Gantt diagram ábrázolása szerkesztő program segítségével vagy Excel programmal.







d.) Határozza meg a processzek végrehajtási sorrendjét!

FCFS	P1	P2	P3	P4	P5
Érkezés	0	1	3	9	12
CPU idő	3	8	2	20	5
Indulás	0	3	11	13	33
Befejezés	3	11	13	33	38
Várakozás	0	2	8	4	21
Végrehajtási sorrend	1	2	3	4	5

SJF

Processz	Érkezési idő	CPU igény	Kezdési idő	Befejezési idő	Várakozási idő	Végrehajtási sorrend
F1	0	3	0	3	0	2
F2	1	5	5	10	4	3
F3	3	2	3	5	0	1
F4	9	5	10	15	1	4
F5	12	5	15	20	3	5

Round Robin					
5 ms	P1	P2	P3	P4	P5
Érkezés	0	1	3	9	12
CPU idő	3	8	2	20	5
Indulás	0	3	8	13	18
Befejezés	3	8,13	10	18,38	23
Várakozás	0	2,2	5	4,5	6
Végrehajtási sorrend	P1	P2	P3	P4	P5

2. feladat

Készítsen egy neptunkod_parent.c és a neptunkod_child.c programokat. A neptunkod_parent.c elindít egy gyermek processzt, ami különbözik a szülőtől. A szülő megvárja a gyermek lefutását. A gyermek szöveget ír a szabványos kimenetre (10-szer) (pl. a hallgató nevét és a neptunkód)! - magyarázza egy-egy mondattal." A fordítás/futtatás után készítsen egy képernyőképet (minden parancs esetén) és illessze be a dokumentumba.

```
[Running] cd "c:\OKGitOS\EYZWG9_0319\" && gcc eyzwg9_child.c -o eyzwg9_child && "c:\OKGitOS\EYZWG9_0319\"eyzwg9_child
Orosz Kristof - EYZWG9
Orosz Kristof - EYZWG9
Orosz Kristof - EYZWG9
Orosz Kristof - EYZWG9
Orosz Kristof - EYZWG9
Orosz Kristof - EYZWG9
Orosz Kristof - EYZWG9
Orosz Kristof - EYZWG9
Orosz Kristof - EYZWG9
Orosz Kristof - EYZWG9
```

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

int main(int argc, char *argv[])
{
    pid_t ppid, pid, cpid;
    ppid = getpid();
    printf("Szulo PID: %d\n", ppid);

    pid = fork();
    if (pid < 0) {
        printf("Fork hiba");
        return 1;
    } else if (pid == 0) {
        cpid = getpid();
        printf("Gyermek PID: %d\n", cpid);
        execl("./eyzwg9_child", "eyzwg9_child", (char *)NULL);
        printf("Execl hiba");
        return 1;
    } else {
        int status;
        if (waitpid(pid, &status, 0) == -1) {
            printf("Waitpid hiba");
            return 1;
        }
        printf("A gyermek futasa befejezodott, visszateresi ertek: %d\n", status);
        return 0;
    }
}
```

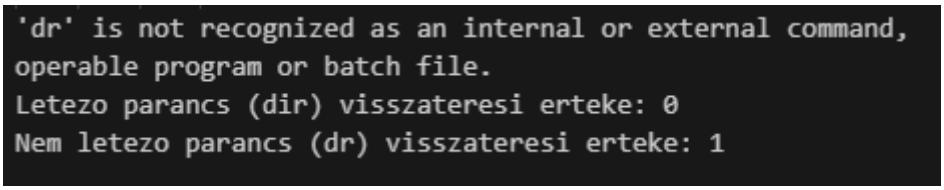
A program először kiírja a szülő folyamat azonosítóját, majd a fork rendszerhívással létrehoz egy új gyermek folyamatot. A gyermek folyamatban kiíródik a saját folyamat azonosítója, és az execl függvénnyel megpróbálja elindítani a "eyzwg9_child" nevű programot. A szülő folyamat a waitpid függvénnyel megvárja, hogy a gyermek folyamat befejeződjön, és ezután kiírja a gyermek visszatérési értékét.

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    for (int i = 0; i < 10; i++) {
        printf("Orosz Kristof - EYZWG9\n");
    }
    return 0;

}
```

Az „eyzwg9_child” nevű programban egy for ciklus tízszer ismétlődik, amely 10 alkalommal fut le. Minden egyes iteráció során a printf függvény segítségével kiírja az "Orosz Kristof - EYZWG9" szöveget a képernyőre.

3. feladat

A system() rendszerhívással hajtson végre létező és nem létező parancsot, és vizsgálja a visszatérési érteket, magyarázza egy-egy mondattal. A fordítás/futtatás után készítsen egy képernyőképet (minden parancs esetén) és illessze be a dokumentumba.




```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    int parancs;
    parancs = system("dir");
    printf("Letezo parancs (dir) visszateresi erteke: %d\n", parancs);
    parancs = system("dr");
    printf("Nem letezo parancs (dr) visszateresi erteke: %d\n", parancs);
    return 0;
}
```

A program először **a system("dir")** parancsot hajtja végre, amely a rendszer könyvtár tartalmát listázza ki, majd kiírja ennek visszatérési értékét a képernyőre. Ezután a **system("dr")** hívás kerül végrehajtásra, ami egy nem létező parancs, és ennek a hívásnak a visszatérési értékét is kiírja.