Operációs rendszerek BSc

5. Gyak.

2022. 03. 07.

Készítette:

Juhász Balázs Bsc

Mérnökinformatikus

ZUYISF

Miskolc, 2022

1. feladat – A ***system()*** rendszerhívással hajtson végre létező és nem létező parancsot, és vizsgálja a visszatérési érteket, magyarázza egy-egy mondattal!
Mentés: *neptunkod1fel.c*

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

int main()
{
    int status;

    if((status = system("date")) < 0)  //éppen aktuális nap dátum
        perror("system() error");      //hiba esetén lefut a perror() függvény

    if(WIFEXITED(status))
        printf("Normalis befejezodes, visszaadott ertek = %d\n", WIFEXITED(status));

}
```

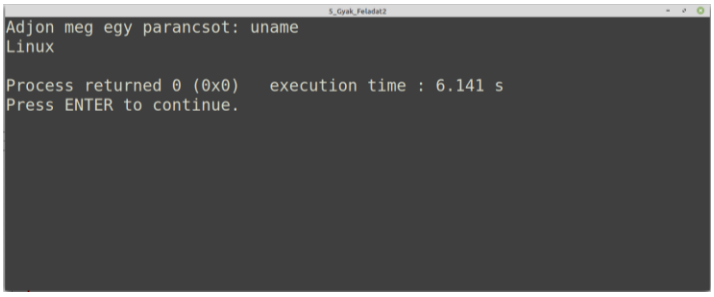
2. feladat – Írjon programot, amely billentyűzetről bekér Unix parancsokat és végrehajtja őket, majd kiírja a szabványos kimenetre. (pl.: amit bekér: date, pwd, who etc.; kilépés: CTRL-\) - magyarázza egy-egy mondattal.
Mentés: *neptunkod2fel.c*

in.c

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

int main()
{
    char input[100];
    printf("Adjon meg egy parancsot: ");
    scanf("%s", input);
    system(input);

    return 0;
}
```



3. feladat – Készítsen egy XY_parent.c és a XY_child.c programokat. A XY_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 neve és a neptunkód)!
Mentés: XY_*parent.c*, ill. *XY_child.c*

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

int main()
{
    pid_t pid;

    if((pid = fork()) < 0)
    {
        perror("fork error");
    }
    else if(pid == 0)
    {
        if(execl("./child", "child", (char *) NULL) < 0)
            perror("execl error");
    }
    if(waitpid(pid, NULL, 0) < 0)
    {
        perror("wait error");
    }

    return 0;
}
```

XY_parent.c

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

int main()
{
    for(int i = 0; i < 10; i++)
    {
        printf("Juhasz Balazs, ZUYISF\n");
        sleep(2);
    }
    return 0;
}
```

XY_child.c

4. feladat – A fork() rendszerhívással hozzon létre egy gyerek processzt-t és abban hívjon meg egy exec családbeli rendszerhívást (pl. execlp). A szülő várja meg a gyerek futását!
Mentés: *neptunkod4fel.c*

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

int main()
{
    pid_t pid;
    if((pid = fork()) < 0)
    {
        perror("fork error");
    }
    else if(pid == 0)
    {
        if(execlp("ls", "-l", "/home/bazsi/Os_Gyak", NULL) < 0)
        {
            perror("execl error");
        }
    }
    if(waitpid(pid, NULL, 0) < 0)
    {
        perror("wait error");
    }

    return 0;
}
```

```
'minta jk_os.pdf'   OS_5_Gyak   _system.c

Process returned 0 (0x0)   execution time : 0.012 s
Press ENTER to continue.
```

5. feladat – A fork() rendszerhívással hozzon létre gyerekeket, várja meg és vizsgálja a befejeződési állapotokat (gyerekben: exit, abort, nullával való osztás)!
Mentés: *neptunkod5fel.c*

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

int main()
{
    pid_t pid, status;

    if((pid = fork()) < 0)
    {
        perror("Hiba a forkban!");
        exit(7);
    }
    else if(pid == 0)
    {
        abort();
        if(wait(&status) != pid)
        {
            perror("Hiba a wait-el!");
        }
    }
    if(WIFEXITED(status))
    {
        printf("Sikeres! :)");
    }

    return 0;
}
```

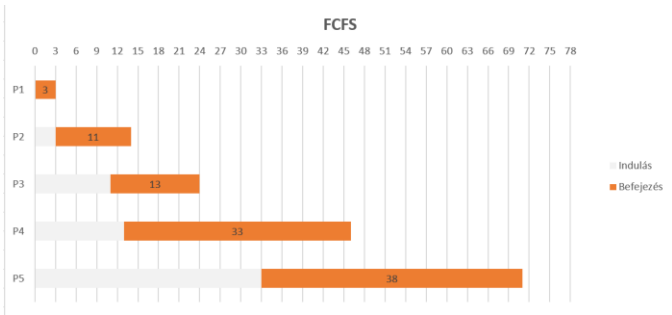
```
Sikeres! :)
Process returned 0 (0x0)   execution time : 0.006 s
Press ENTER to continue.
```

6. feladat – Határozza meg FCFS és SJF esetén
a.) A befejezési időt?
b.) A várakozási/átlagos várakozási időt?
c.) Ábrázolja Gantt diagram segítségével *az aktív/várakozó processzek* futásának menetét.
Megj.: a Gantt diagram ábrázolása szerkesztő program segítségével vagy Excel programmal.
Mentés: *neptunkod6fel.pdf*

FCFS megoldás:

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

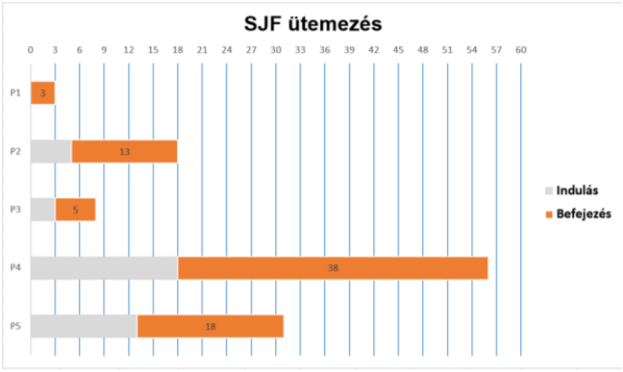
Diagram:



SJF megoldás:

SJF	Érkezés	CPU idő	Indulás	Befejezés	Várakozás
P1	0	3	0	3	0
P2	1	8	5	13	4
P3	3	2	3	5	0
P4	9	20	18	38	9
P5	12	5	13	18	1

Diagram:



Round Robin (RR) esetén

a.) Ütemezze az adott időszelet (5ms) alapján az egyes processzek (befejezési és várakozási/átlagos várakozási idő) paramétereit (ms)!
b.) A rendszerben lévő processzek végrehajtásának sorrendjét?
c.) Ábrázolja Gantt diagram segítségével az *aktív/várakozó processzek* futásának menetét!"
Megj.: a Gantt diagram ábrázolása szerkesztő program segítségével vagy Excel programmal.
Mentés: *neptunkod6fel pdf*

Round Robin megoldás:

RR: 5ms	Érkezés	CPU idő	Indulás	Befejezés	Várakozás	Várakozó processz
P1	0	3	0	**3**	**0**	**P2**
P2	1	8	3	**8**	**2**	**P2, P3**
P3	3	2	8	**10**	**5**	**P2, P4**
P4	9	20	13	**18**	**4**	**P4, P5**
P5	12	5	18	**23**	**6**	**P4**

Diagram: