

Operációs rendszerek BSc

5. Gyak.

2022. 03. 07.

Készítette:

Szkárosi Szilárd Bsc

Mérnökinformatikus

DLWGQZ

Miskolc, 2022

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

Mentés: *neptunkod1fel.c*

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <sys/types.h>
4  #include <sys/wait.h>
5
6  int main()
7  {
8      int status;
9
10     if((status = system("date")) < 0) //éppen aktuális nap dátum
11         perror("system() error"); //hiba esetén lefut a perror() függvény
12
13     if(WIFEXITED(status))
14         printf("Normalis befejezodes, visszaadott ertek = %d\n", WIFEXITED(status));
15
16 }
17

```

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*

```

#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;
}

```

```

5. Gyak. Feladat2
Adjon meg egy parancsot: uname
Linux

Process returned 0 (0x0)   execution time : 6.141 s
Press ENTER to continue.

```

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*

```

XY_parent.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <sys/wait.h>
4  #include <sys/types.h>
5
6  int main()
7  {
8      pid_t pid;
9
10     if((pid = fork()) < 0)
11     {
12         perror("fork error");
13     }
14     else if(pid == 0)
15     {
16         if(execl("./child", "child", (char *) NULL) < 0)
17             perror("execl error");
18     }
19     if(waitpid(pid, NULL, 0) < 0)
20     {
21         perror("wait error");
22     }
23
24     return 0;
25 }
26

```

XY_parent.c

```

XY_child.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <sys/wait.h>
4  #include <sys/types.h>
5  #include <unistd.h>
6
7  int main()
8  {
9      for(int i = 0; i < 10; i++)
10     {
11         printf("Szkárosi Szilárd, DLWGQZ\n");
12         sleep(2);
13     }
14
15     return 0;
16 }
17

```

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*

```

DLWGQZ4fel.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <sys/wait.h>
4  #include <sys/types.h>
5  #include <unistd.h>
6
7  int main()
8  {
9      pid_t pid;
10
11     if((pid = fork()) < 0)
12     {
13         perror("fork error");
14     }
15     else if(pid == 0)
16     {
17         if(execlp("ls", "-l", "/home/szkarosil/05_GYAK", NULL) < 0)
18             perror("execl error");
19     }
20     if(waitpid(pid, NULL, 0) < 0)
21     {
22         perror("wait error");
23     }
24
25     return 0;
26 }
27
28

```

```
'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 (gyerekekben: `exit`, `abort`, nullával való osztás)!

Mentés: *neptunkod5fel.c*

```
DLWQZ5fel.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <sys/wait.h>
4  #include <sys/types.h>
5  #include <unistd.h>
6
7  int main()
8  {
9      pid_t pid, status;
10
11      if((pid = fork()) < 0)
12      {
13          perror("Hiba a forkban!");
14          exit(7);
15      }
16      else if(pid == 0)
17      {
18          abort();
19          if(wait(&status) != pid)
20          {
21              perror("Hiba a wait-el!");
22          }
23      }
24      if(WIFEXITED(status))
25      {
26          printf("Sikeres! :)\n");
27      }
28      return 0;
29  }
```

```
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.

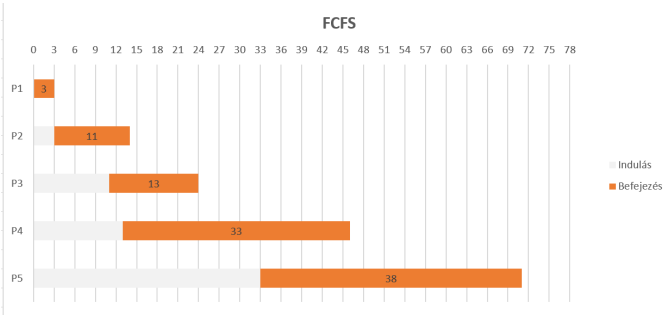
Megj.: a Gantt diagram ábrázolása szerkesztő program segítségével vagy Excel programmal.

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

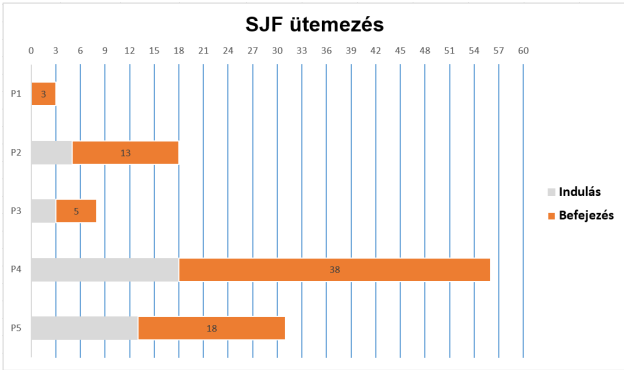
Diagram:



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 időszel (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:

