

# **Operációs rendszerek BSc**

**5. Gyak.**

**2022. 03. 09.**

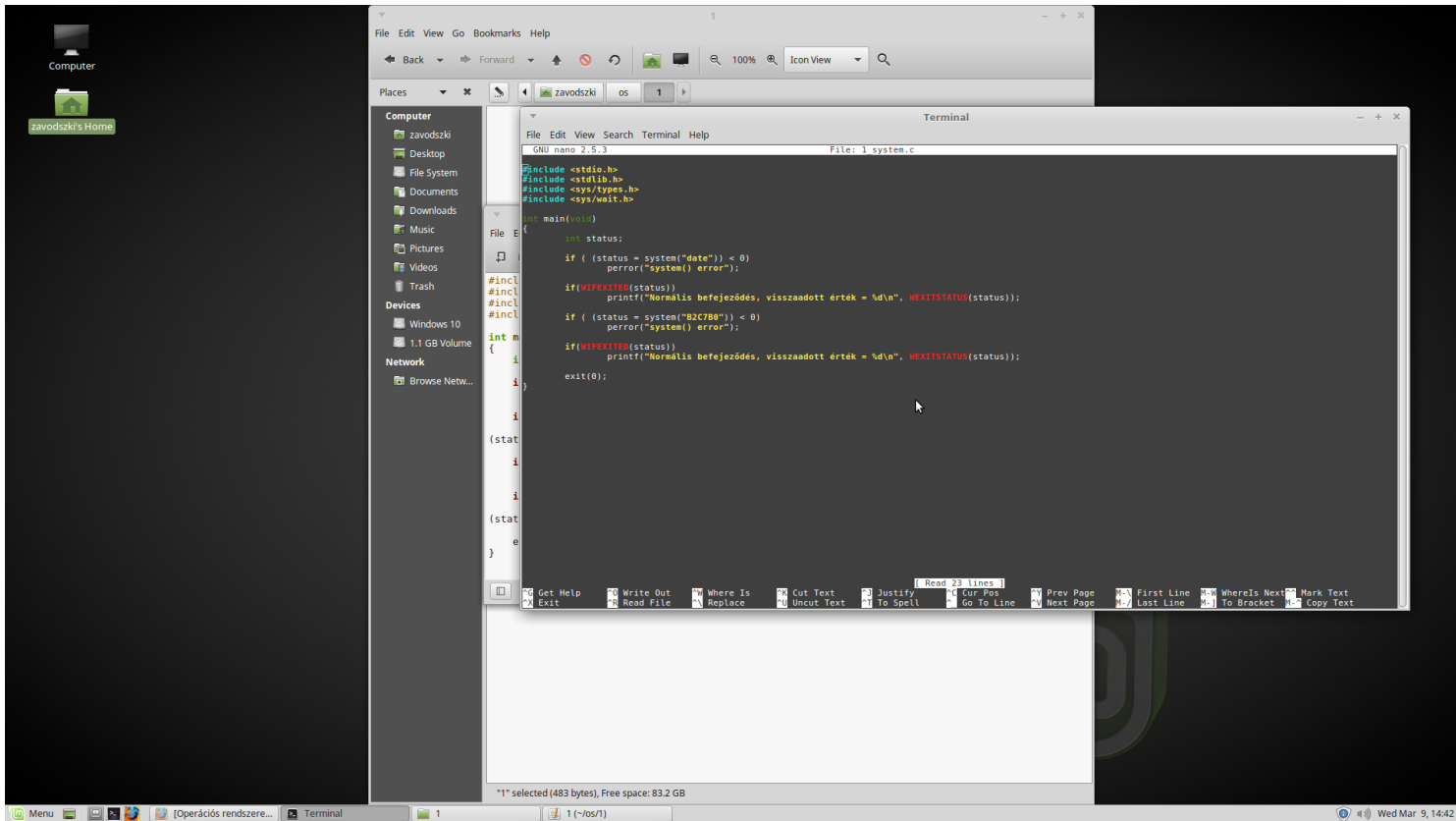
**Készítette:**

Závodszki Máté  
Mérnökinformatikus  
B2C7B0

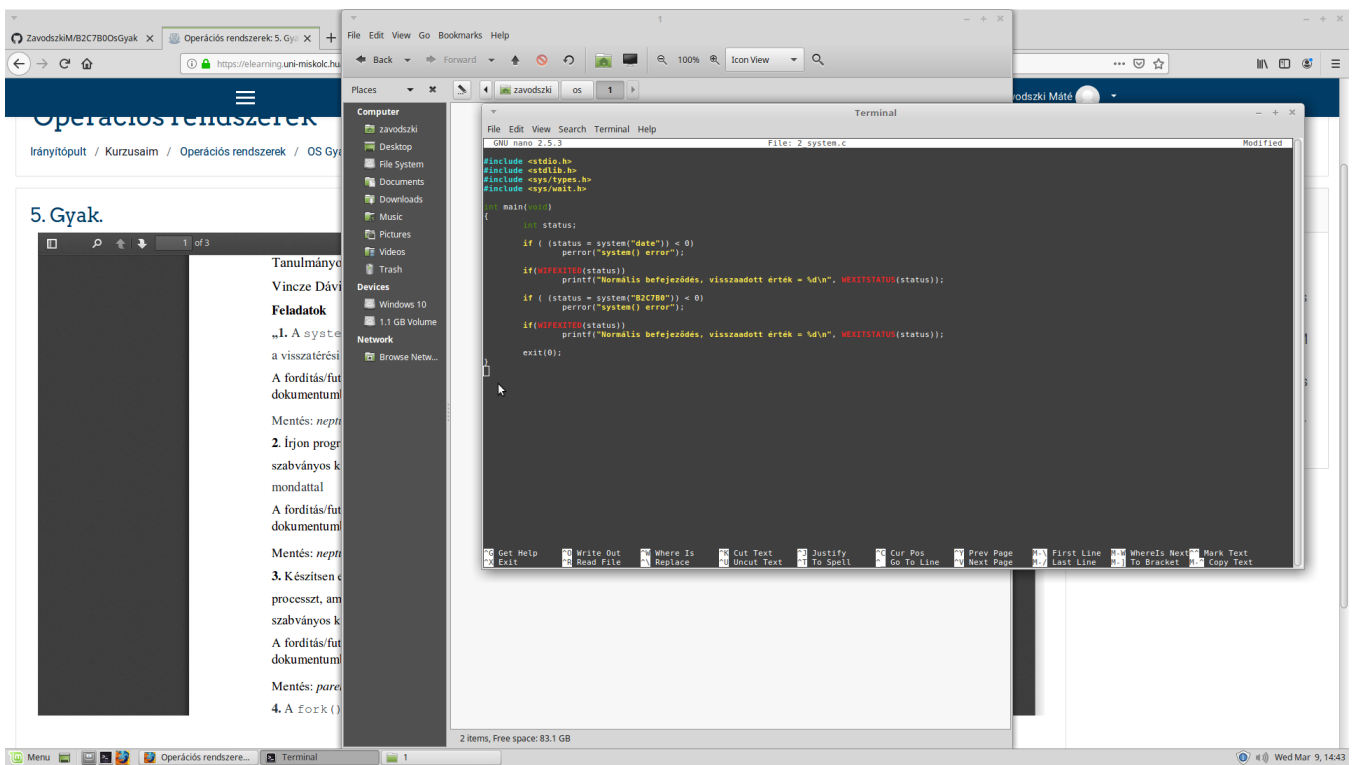
**Miskolc, 2022**

## A feladat

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



2, Írjon programot, amely billentyűzetről bekér Unix parancsokat és végrehajtja őket, majd kiírja a szabványos kimenetre.



3, Készítsen egy `parent.c` és a `child.c` programokat. A `parent.c` elindít egy gyermek processzt, ami különbözik a szülőtől. A szülő megvárja a gyermek lefutását. A gyermek szöveget ír a szabványos kimenetre (10-ször) (pl. a hallgató neve és a neptunkód)!

The screenshot shows a web browser window with the URL <https://elearning.uni-miskolc.hu/zart/mod/resource/view.php?id=71148>. The page title is "Operációs rendszerek" and the breadcrumb is "Irányítópult / Kursusaim / Operációs rendszerek / OS Gyakorlat / 5. Gyak.". The main content is titled "5. Gyak." and contains a list of tasks. The first task is to create a program that prints the user's name and neptunkód 10 times. The second task is to create a program that prints the user's name and neptunkód 10 times. The third task is to create a `parent.c` and a `child.c` program. The `parent.c` program should create a child process using `fork()` and wait for it to finish. The `child.c` program should print the user's name and neptunkód 10 times. The terminal window shows the code for `child.c`:

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

int main(void)
{
    for (i = 0; i < 10; i++)
    {
        printf("Zavodszki Máté, B2C7B00");
    }
    exit(0);
}
```

The screenshot shows the same web browser window as above. The terminal window now shows the code for `parent.c`:

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

int main(void)
{
    pid_t waitpid(pid_t pid, int *status);
    pid_t waitpid(pid_t pid, int *status);

    pid_t waitpid(pid_t pid, int *status);
    pid_t waitpid(pid_t pid, int *status);

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

4, 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!

The screenshot shows a web browser window with the URL <https://elearning.uni-miskolc.hu/zart/mod/resource/view.php?id=71148>. The page is titled "Operációs rendszerek" and contains a document titled "5. Gyak.". The document text is as follows:

A fordítás/futtatás után készítsen egy képernyőképét a dokumentumba.  
Mentés: `neptunkod1fel.c`

2. Írjon programot, amely billentyűzetről bekér U...  
szabványos kimenetre. (pl.: amit bekér: date, pwd...  
mondattal)  
A fordítás/futtatás után készítsen egy képernyőképét a dokumentumba.  
Mentés: `neptunkod2fel.c`

3. Készítsen egy `parent.c` és a `child.c` pro...  
processzt, ami különbözik a szülőtől. A szülő me...  
szabványos kimenetre (10-ször) (pl. a hallgató ne...  
A fordítás/futtatás után készítsen egy képernyőképét a dokumentumba.  
Mentés: `parent.c`, ill. `child.c`

4. A `fork()` rendszerhívással hozzon létre egy...  
családbeli rendszerhívást (pl. `execlp`). A szülő várja meg a gyerek futását! - magyarázza egy-egy...  
mondattal.  
A fordítás/futtatás után készítsen egy képernyőképét (minden parancs esetén) és illessze be a dokumentumba.  
Mentés: `neptunkod4fel.c`

The terminal window shows the following C code:

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

int main(void)
{
    if ((pid = fork()) < 0)
        perror("fork error");
    else if (pid == 0)
        execl(7);
    if (wait(&status) != pid)
        perror("wait hiba");
    if (WIFEXITED(status))
        printf("Normális befejeződés, visszaadott érték = %d\n", WEXITSTATUS(status));

    if ((pid = fork()) < 0)
        perror("fork hiba");
    else if (pid == 0)
        abort();
    if (wait(&status) != pid)
        perror("wait hiba");
    if (WIFEXITED(status))
        printf("Abnormális befejeződés a szignál sorszáma = %d\n", WTERMSIG(status));
}
```

5, A `fork()` rendszerhívással hozzon létre gyerekeket, várja meg és vizsgálja a befejeződési állapotokat (gyerekekben: `exit`, `abort`, nullával való osztás)!

The screenshot shows a web browser window with the URL <https://elearning.uni-miskolc.hu/zart/mod/resource/view.php?id=71148>. The page is titled "Operációs rendszerek" and contains a document titled "5. Gyak.". The document text is as follows:

A fordítás/futtatás után készítsen egy képernyőképét a dokumentumba.  
Mentés: `neptunkod1fel.c`

2. Írjon programot, amely billentyűzetről bekér U...  
szabványos kimenetre. (pl.: amit bekér: date, pwd...  
mondattal)  
A fordítás/futtatás után készítsen egy képernyőképét a dokumentumba.  
Mentés: `neptunkod2fel.c`

3. Készítsen egy `parent.c` és a `child.c` pro...  
processzt, ami különbözik a szülőtől. A szülő me...  
szabványos kimenetre (10-ször) (pl. a hallgató ne...  
A fordítás/futtatás után készítsen egy képernyőképét a dokumentumba.  
Mentés: `parent.c`, ill. `child.c`

4. A `fork()` rendszerhívással hozzon létre egy...  
családbeli rendszerhívást (pl. `execlp`). A szülő várja meg a gyerek futását! - magyarázza egy-egy...  
mondattal.  
A fordítás/futtatás után készítsen egy képernyőképét (minden parancs esetén) és illessze be a dokumentumba.  
Mentés: `neptunkod4fel.c`

The terminal window shows the following C code:

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

void pr_exit(int status)
{
    if (WIFEXITED(status))
        printf("Normális befejeződés a szignál sorszáma = %d\n", WEXITSTATUS(status));
    else
        printf("Abnormális befejeződés a szignál sorszáma = %d\n", WTERMSIG(status));
}

int main(void)
{
    pid_t pid;
    int status;

    if ((pid = fork()) < 0)
        perror("fork error");
    else if (pid == 0)
        exit(7);
    if (wait(&status) != pid)
        perror("wait hiba");
    pr_exit(status);

    if ((pid = fork()) < 0)
        perror("fork error");
    else if (pid == 0)
        abort();
    if (wait(&status) != pid)
        perror("wait hiba");
    pr_exit(status);

    if ((pid = fork()) < 0)
        perror("fork hiba");
    else if (pid == 0)
        abort();
    if (wait(&status) != pid)
        perror("wait hiba");
}
```

B feladat – FCFS, SJF, Round Robin

- a.) A befejezési időt?
- b.) A várakozási/átlagos várakozási időt?
- c.) Ábrázolja Gantt diagram segítségével az *aktív/várakozó processzek* futásának menetét.



## FCFS II

Válassza ki a kiemelendő időszakot a jobb oldalon. A diagram jelmagyarázata következik.

Kiemelt időszak: 1

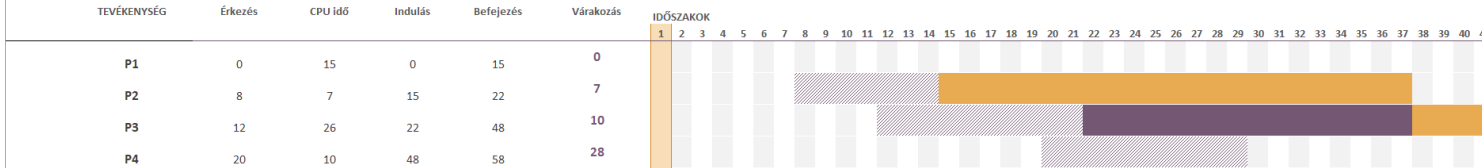
Érkezés

Indulás

% kész

Tényleges (a terven felül)

% kész (a terven felül)



## SJF II

Válassza ki a kiemelendő időszakot a jobb oldalon. A diagram jelmagyarázata következik.

Kiemelt időszak: 1

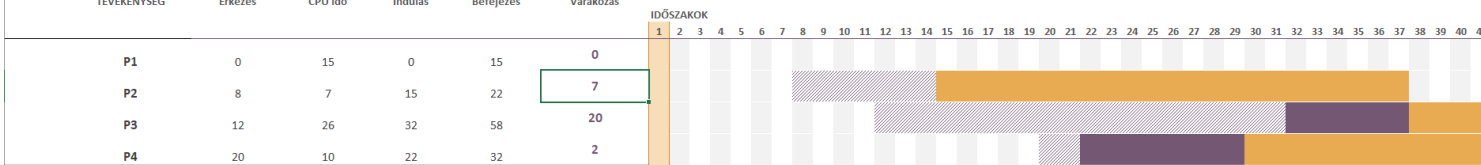
Érkezés

Indulás

% kész

Tényleges (a terven felül)

% kész (a terven felül)



## Round Robin II

Válassza ki a kiemelendő időszakot a jobb oldalon. A diagram jelmagyarázata következik.

Kiemelt időszak: 1

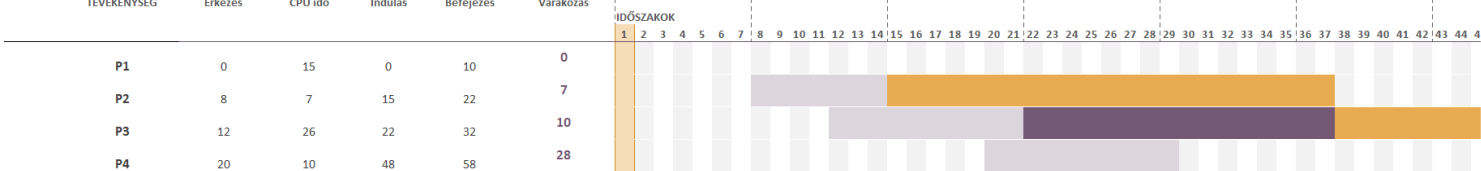
Érkezés

Indulás

% kész

Tényleges (a terven felül)

% kész (a terven felül)



## Round Robin III

Válassza ki a kiemelendő időszakot a jobb oldalon. A diagram jelmagyarázata következik.

Kiemelt időszak: 1

Érkezés

Indulás

% kész

Tényleges (a terven felül)

% kész (a terven felül)

