

Operációs rendszerek BSc

5. Gyak.

2022. 03. 09.

Készítette:

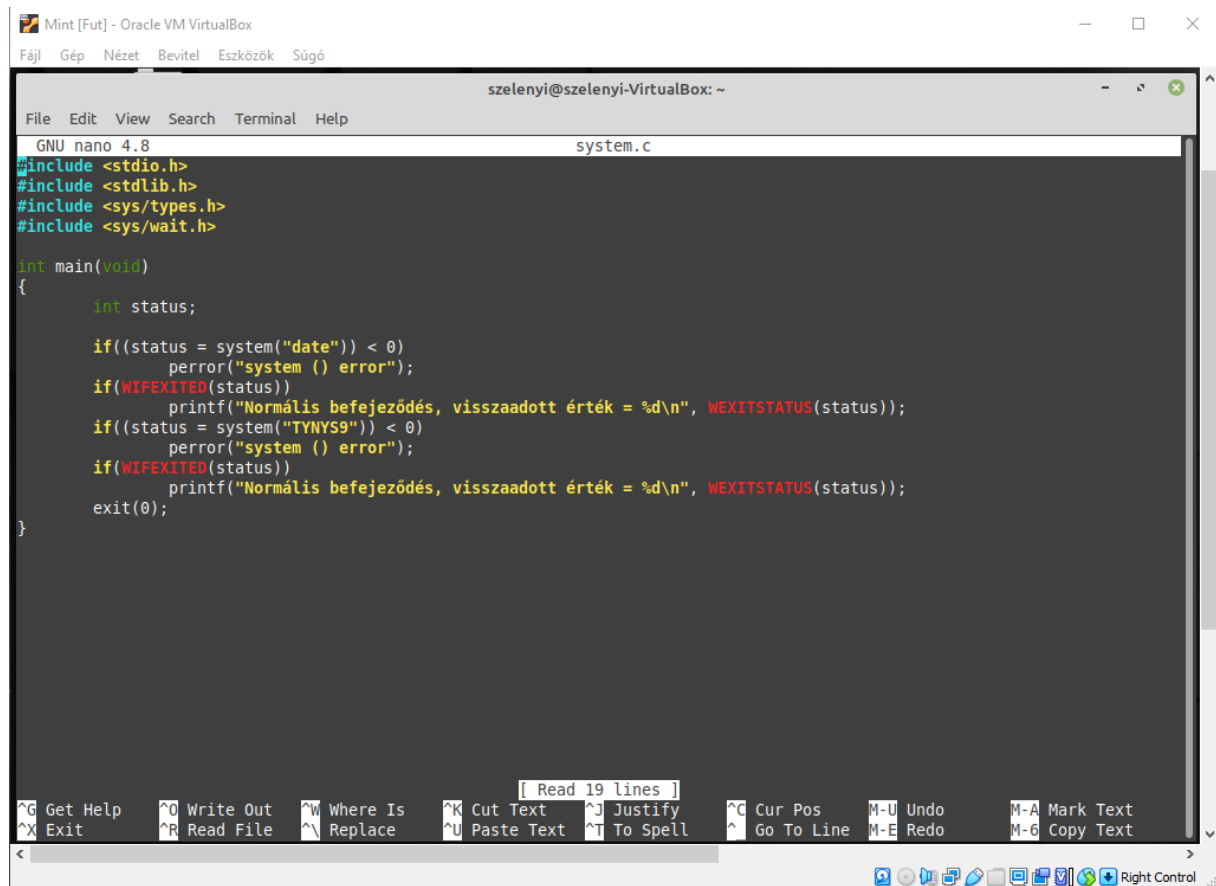
Szelényi Szabolcs Bsc

Mérnökinformatikus hallgató

TYNYS9

Miskolc, 2022

1. A `system()` rendszerhívással hajtson végre létező és nem létező parancsot, és vizsgálja a visszatérési értéket



The screenshot shows a nano 4.8 editor window titled 'system.c' inside a VirtualBox machine named 'szelenyi@szelenyi-VirtualBox: ~'. The code is as follows:

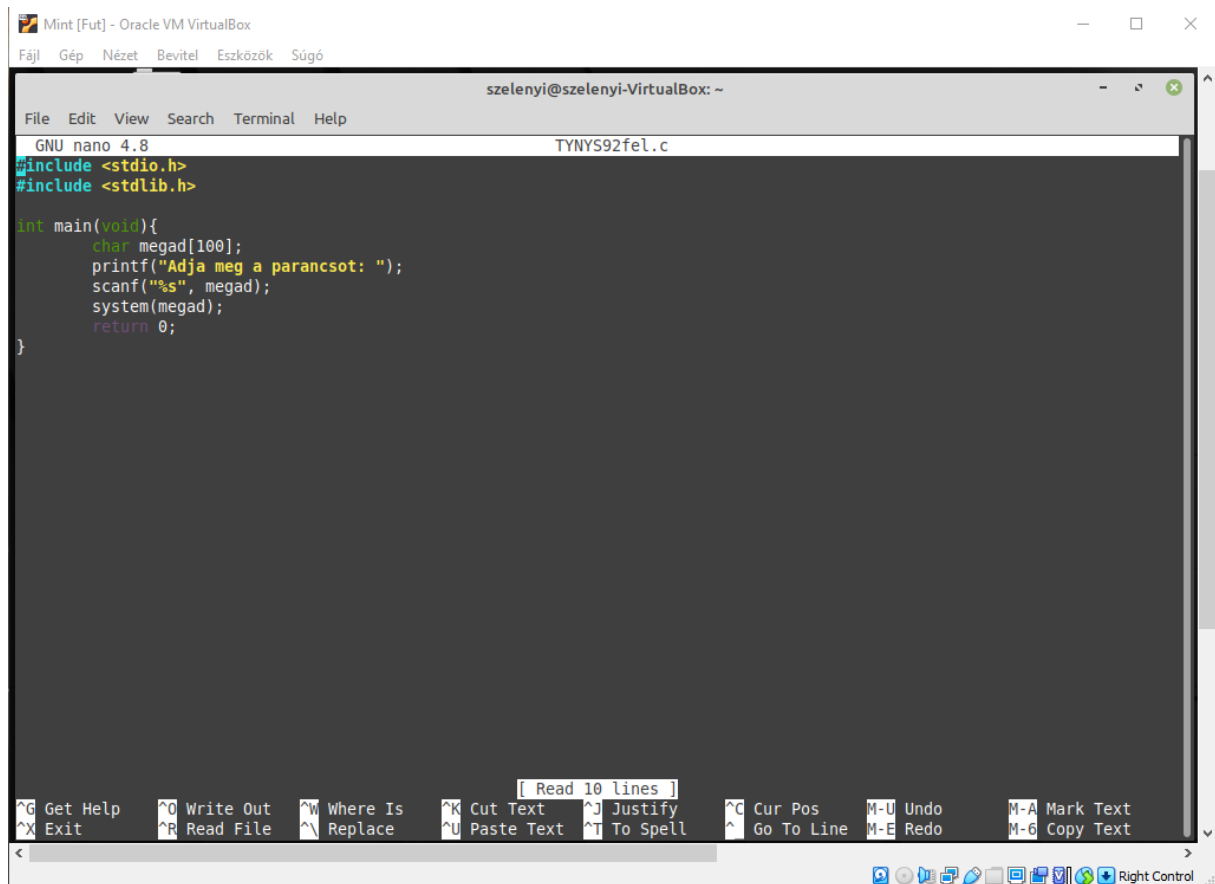
```
GNU nano 4.8 system.c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>

int main(void)
{
    int status;

    if((status = system("date")) < 0)
        perror("system () error");
    if(WIFEXITED(status))
        printf("Normális befejeződés, visszaadott érték = %d\n", WEXITSTATUS(status));
    if((status = system("TINY9")) < 0)
        perror("system () error");
    if(WIFEXITED(status))
        printf("Normális befejeződés, visszaadott érték = %d\n", WEXITSTATUS(status));
    exit(0);
}
```

The bottom of the window displays a status bar with various keyboard shortcuts and a 'Right Control' button.

2. Írjon programot, amely billentyűzetről bekér Unix parancsokat és végrehajtja őket, majd kiírja a szabványos kimenetre. (pl.: amit bekér: date, pwd, who etc.; kilépés: CTRL)



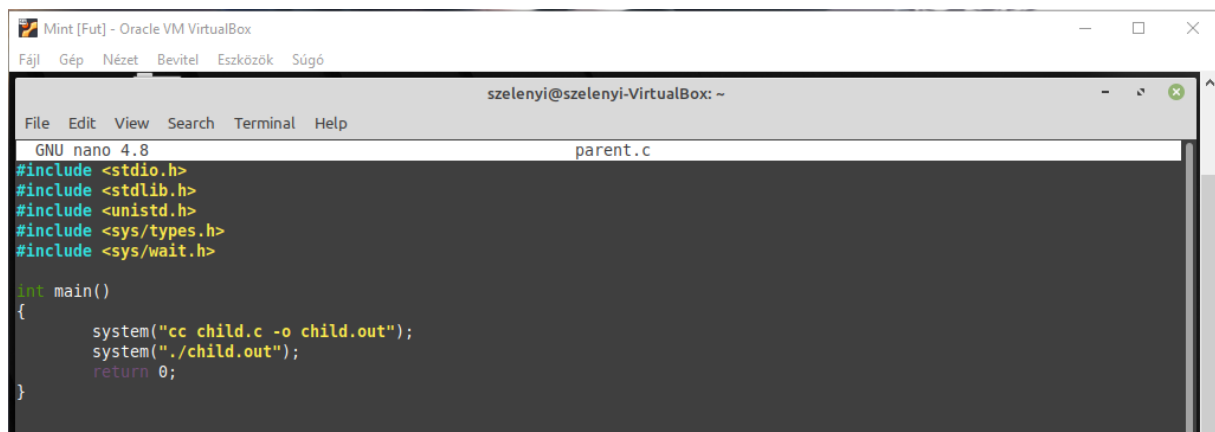
The screenshot shows a nano editor window titled 'szelenyi@szelenyi-VirtualBox: ~' editing a file named 'TYNYS92fel.c'. The code is as follows:

```
GNU nano 4.8 TYNYS92fel.c
#include <stdio.h>
#include <stdlib.h>

int main(void){
    char megad[100];
    printf("Adja meg a parancsot: ");
    scanf("%s", megad);
    system(megad);
    return 0;
}
```

The bottom status bar shows various keyboard shortcuts: ^G Get Help, ^O Write Out, ^W Where Is, ^K Cut Text, ^J Justify, ^C Cur Pos, M-U Undo, M-A Mark Text, ^X Exit, ^R Read File, ^M Replace, ^U Paste Text, ^T To Spell, ^_ Go To Line, M-E Redo, M-6 Copy Text. A '[Read 10 lines]' indicator is also present.

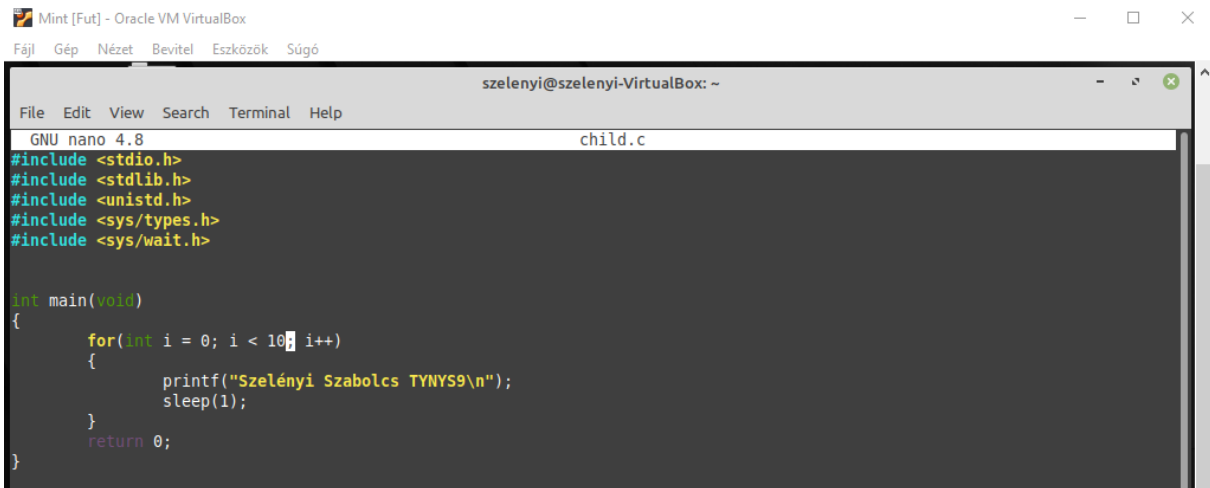
3. Készítsen egy parent.c és a child.c programokat. A parent.c elindít egy gyermek processzt, ami különbözik a szülőtől. A szülő megvárja a gyermek lefutását. A gyermek szöveget ír a szabványos kimenetre (10-ször) (pl. a hallgató neve és a neptunkód)!



The screenshot shows a nano editor window titled 'szelenyi@szelenyi-VirtualBox: ~' editing a file named 'parent.c'. The code is as follows:

```
GNU nano 4.8 parent.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

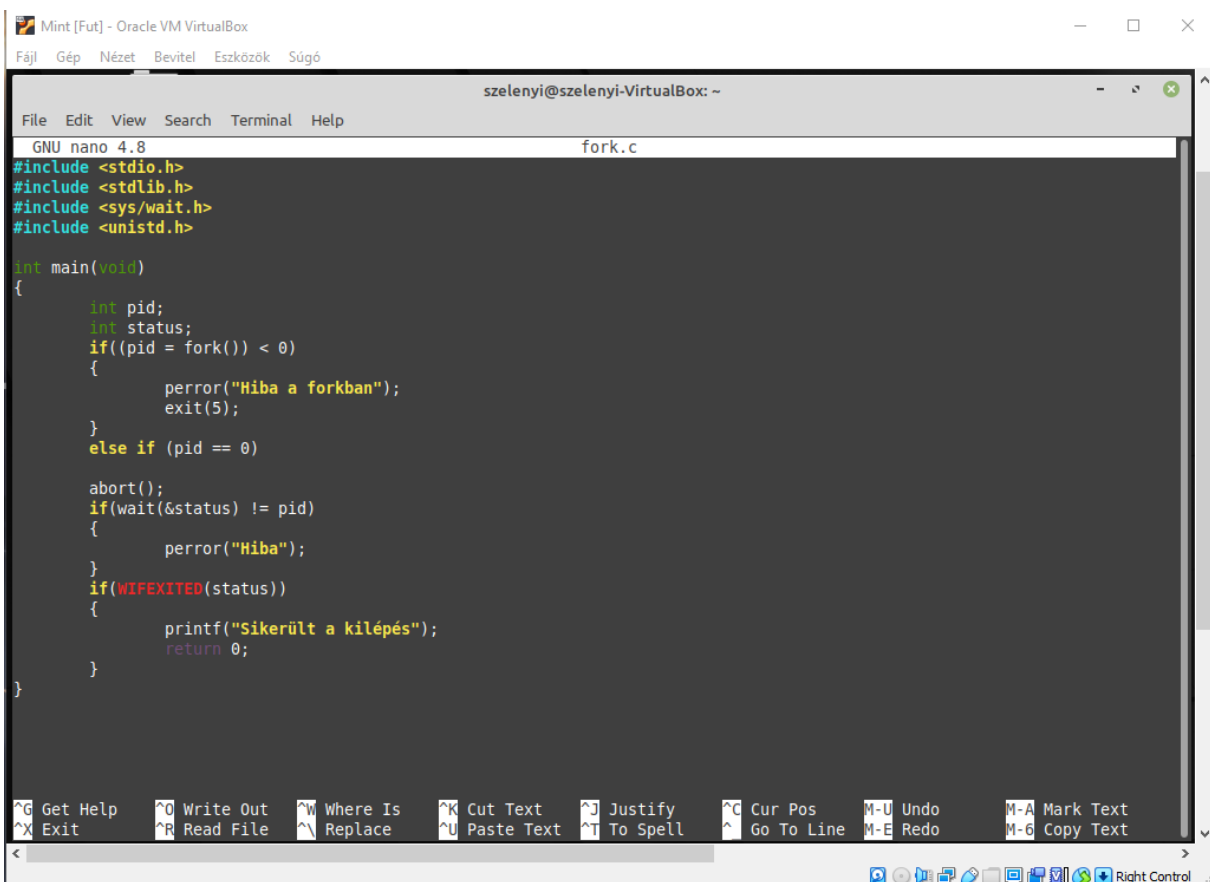
int main()
{
    system("cc child.c -o child.out");
    system("./child.out");
    return 0;
}
```



```
GNU nano 4.8 child.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int main(void)
{
    for(int i = 0; i < 10; i++)
    {
        printf("Szelényi Szabolcs TYNYS9\n");
        sleep(1);
    }
    return 0;
}
```

4. A fork() rendszerhívással hozzon létre egy gyerek processzt és abban hívjon meg egy exec családbeli rendszerhívást (pl. execlp). A szülő várja meg a gyerek futását!



```
GNU nano 4.8 fork.c
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <unistd.h>

int main(void)
{
    int pid;
    int status;
    if((pid = fork()) < 0)
    {
        perror("Hiba a forkban");
        exit(5);
    }
    else if (pid == 0)
    {
        abort();
        if(wait(&status) != pid)
        {
            perror("Hiba");
        }
        if(WIFEXITED(status))
        {
            printf("Sikerült a kilépés");
            return 0;
        }
    }
}
```

5. A fork() rendszerhívással hozzon létre gyerekeket, várja meg és vizsgálja a befejeződési állapotokat (gyerekekben: exit, abort, nullával való osztás)! - magyarázza egy-egy mondattal! A fordítás/futtatás után készítsen egy képernyőképet (minden parancs esetén) és illessze be a dokumentumba.

```

szelenyi@szelenyi-VirtualBox: ~
File Edit View Search Terminal Help
GNU nano 4.8 TYNYS95fel.c Modified
#include <stdlib.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <unistd.h>

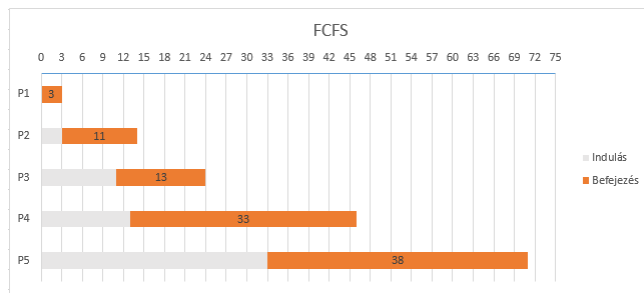
int main()
{
    pid_t pid, status;
    if((pid = fork()) < 0){
        perror("Hiba a forkban!");
        exit(7);
    }
    else if(pid == 0){
        abort();
        if(wait(&status) != pid){
            perror("Hiba a wait-el!");
        }
    }
    if(WIFEXITED(status)){
        printf("Sikeress");
    }
    return 0;
}

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify
^X Exit      ^R Read File  ^\ Replace   ^U Paste Text ^T To Spell

```

6. Adott a következő ütemezési feladat, amit a FCFS, SJF és Round Robin (RR) ütemezési algoritmus használatával készítsen el (külön-külön táblázatba): I. 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.

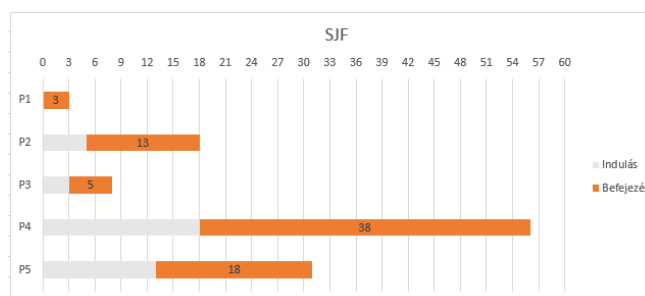
FCFS						
	FCFS	Érkezés	CPU idő			
	P1	0	3			
	P2	1	8			
	P3	3	2			
	P4	9	20			
	P5	12	5			
	FCFS	Érkezés	CPU idő	Indulás	Befejezés	Várakozás
	P1	0	3	0	3	0
	P2	1	8	3	11	2
	P3	3	2	11	13	8
	P4	9	20	13	33	4
	P5	12	5	33	38	21



SJF

SJF	Érkezés	CPU idő			
P1	0	3			
P2	1	8			
P3	3	2			
P4	9	20			
P5	12	5			

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



II. Round Robin (RR) esetén a.) Ütemezze az adott időszelét (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.

RR	Érkezés	CPU idő				
P1	0	3				
P2	1	8				
P3	3	2				
P4	9	20				
P5	12	5				

RR	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

