

Operációs rendszerek BSc

5.Gyak.

2022. 03. 08.

Készítette:

Ónodi Bence BSC

Programtervező
informatikus

RYSNLC

Miskolc, 2022

1. A system() rendszerhívással hajtson végre létező és nem létező parancsot, és vizsgálja

a visszatérési értéket, magyarázza egy-egy mondattal

```
Volume in drive C has no label.
Volume Serial Number is 22C8-FA62

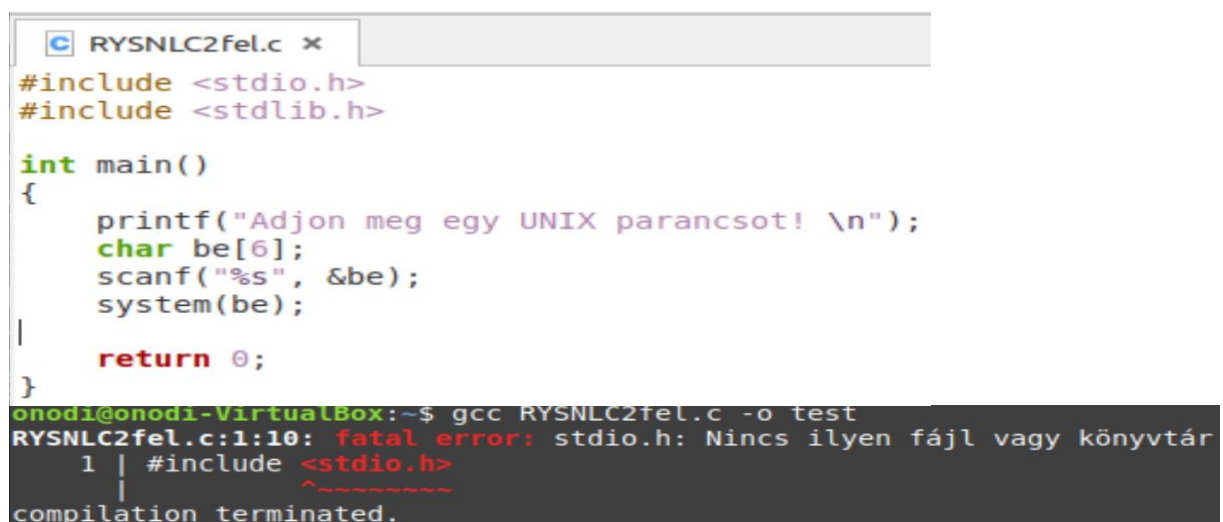
Directory of C:\Users\onodi\Desktop\Ónodi Bence\Uni\2022 II.félév\OS\5.gyak\neptunkod1fel

2022. 03. 08. 15:09 <DIR>      .
2022. 03. 08. 15:09 <DIR>      ..
2022. 03. 08. 15:08 <DIR>      bin
2022. 03. 08. 15:09          248 main.c
2022. 03. 08. 15:08          1 074 neptunkod1fel.cbp
2022. 03. 08. 15:08 <DIR>      obj
                2 File(s)        1 322 bytes
                4 Dir(s)  363 875 758 080 bytes free

Korte!
Korte!
Korte!
'Nincs' is not recognized as an internal or external command,
operable program or batch file.

Process returned 0 (0x0)   execution time : 0.144 s
Press any key to continue.
```

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 image shows a code editor window titled 'RYSNLC2fel.c' containing a C program. The program includes `<stdio.h>` and `<stdlib.h>`, and has a `main` function that prints a message, reads a command from the user, and attempts to execute it using `system`. Below the code, a terminal window shows the compilation command `gcc RYSNLC2fel.c -o test` and a fatal error message: `RYSNLC2fel.c:1:10: fatal error: stdio.h: Nincs ilyen fájl vagy könyvtár`. The error points to the `#include <stdio.h>` line in the code.

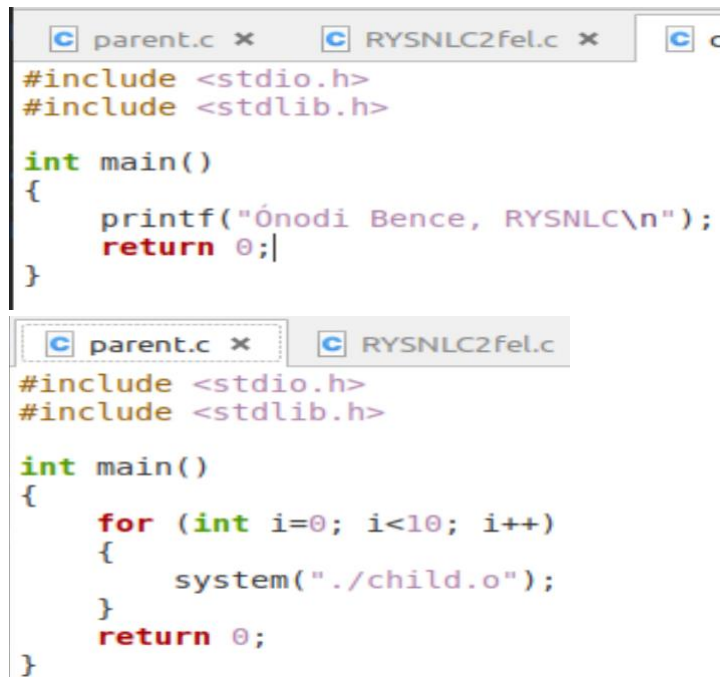
```
RYSNLC2fel.c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("Adjon meg egy UNIX parancsot! \n");
    char be[6];
    scanf("%s", &be);
    system(be);
    return 0;
}

onodi@onodi-VirtualBox:~$ gcc RYSNLC2fel.c -o test
RYSNLC2fel.c:1:10: fatal error: stdio.h: Nincs ilyen fájl vagy könyvtár
1 | #include <stdio.h>
  | ^~~~~~
compilation terminated.
```

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)!



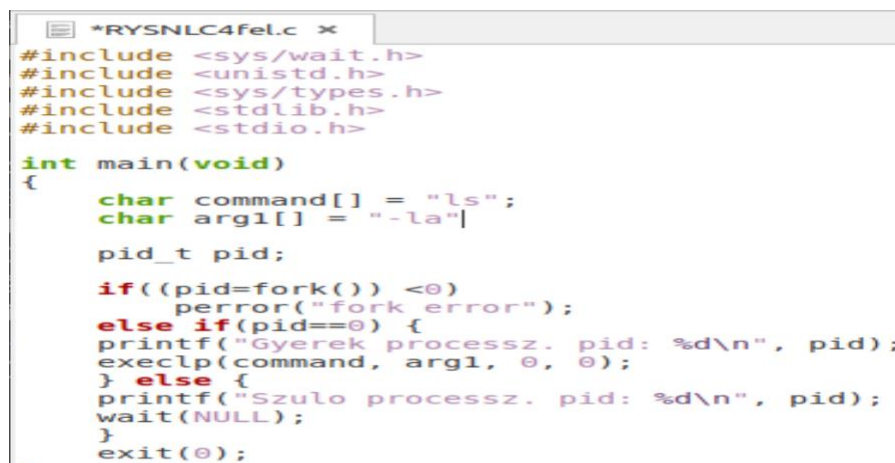
```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("Ónodi Bence, RYSNLC\n");
    return 0;
}
```

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

int main()
{
    for (int i=0; i<10; i++)
    {
        system("./child.o");
    }
    return 0;
}
```

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!



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

int main(void)
{
    char command[] = "ls";
    char arg1[] = "-la";

    pid_t pid;

    if((pid=fork()) <0)
        perror("fork error");
    else if(pid==0) {
        printf("Gyerek processz. pid: %d\n", pid);
        execlp(command, arg1, 0, 0);
    } else {
        printf("Szulo processz. pid: %d\n", pid);
        wait(NULL);
    }
    exit(0);
}
```

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

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

int main()
{
    pid_t p = fork();
    pid_t p2 = fork();

    if(p==-1)
    {
        perror("fork failed");
        return EXIT_FAILURE;
    }
    else if(p==0)
    {
        execl("/bin/sh", "bin/sh" "-c", "./error", "NULL");
        return EXIT_FAILURE;
    }

    if(p2==-1)
    {
        perror("fork failed");
        return EXIT_FAILURE;
    }
    else if(p2==0)
    {
        execl("/bin/sh", "bin/sh" "-c", "./dividezero", "NULL");
        return EXIT_FAILURE;
    }

    if(WIFEXITED(status))
    {
        const int es = WEXITSTATUS(status);
        printf("Exit status was %d\n", es);
    }

    if(waitpid(p2, &status, 1) == -1)
    {
        perror("waitpid failed");
        return EXIT_FAILURE;
    }
}
```

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.

Mentés: neptunkod6fel pdf

II. Round Robin (RR) esetén

a.) Ütemezze az adott időszeklet (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.