

# JEGYZŐKÖNYV

Operációs rendszerek BSc

2022. tavasz féléves feladat

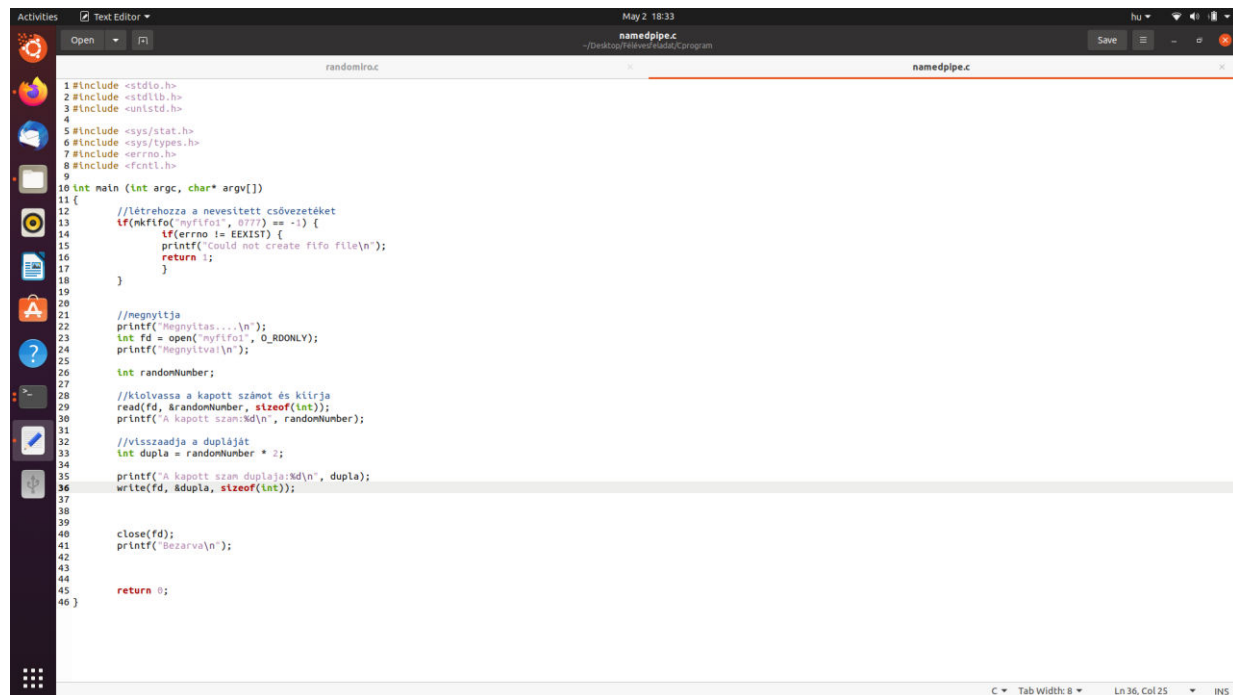
Készítette: Boján Miron Noel

Neptunkód: F4XQRO

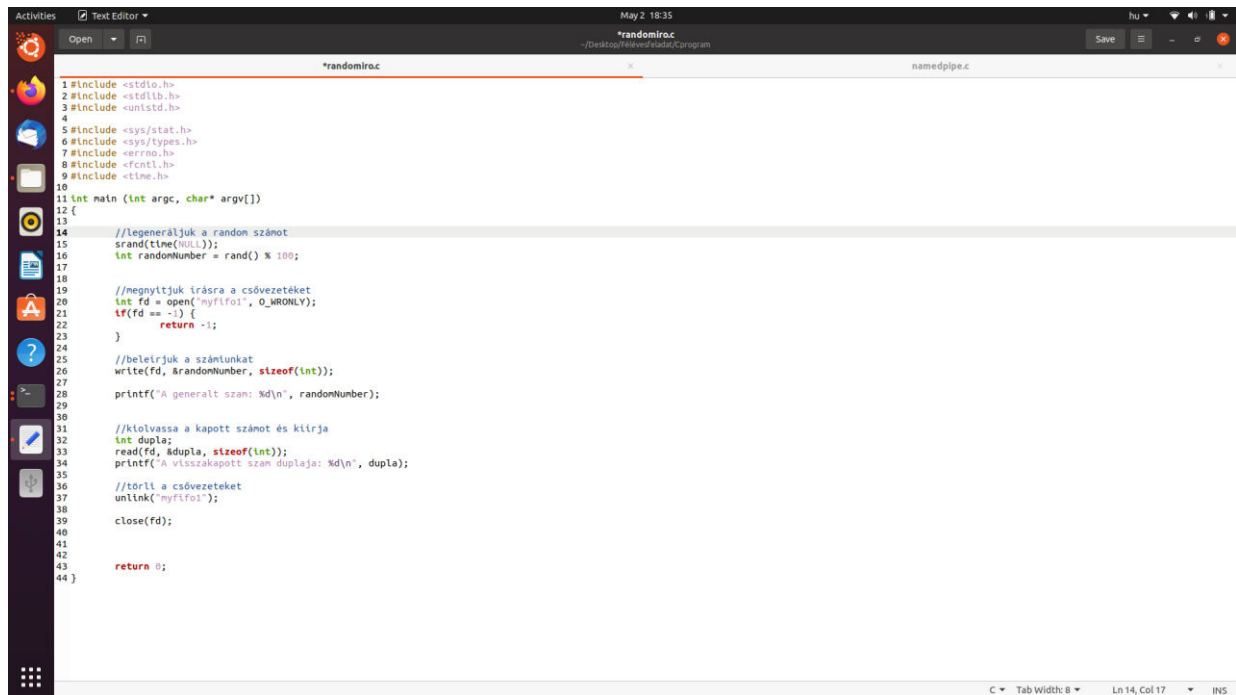
## 1. A feladat leírása:

6. Írjon C nyelvű programokat, ami létrehoz egy nevesített csövezeteket (bejegyzes az fs-en) megnyitja olvasni próbál belőle közben egy másik program ír bele egy random számot az első program kiírja a kapott számot és visszaküldi ennek a dupláját a másik program kiolvassa és kiírja a képernyőre a processzek megszűnnek és a második program eltünteti a nevesített csövezeteket

## A feladat elkészítésének lépései:

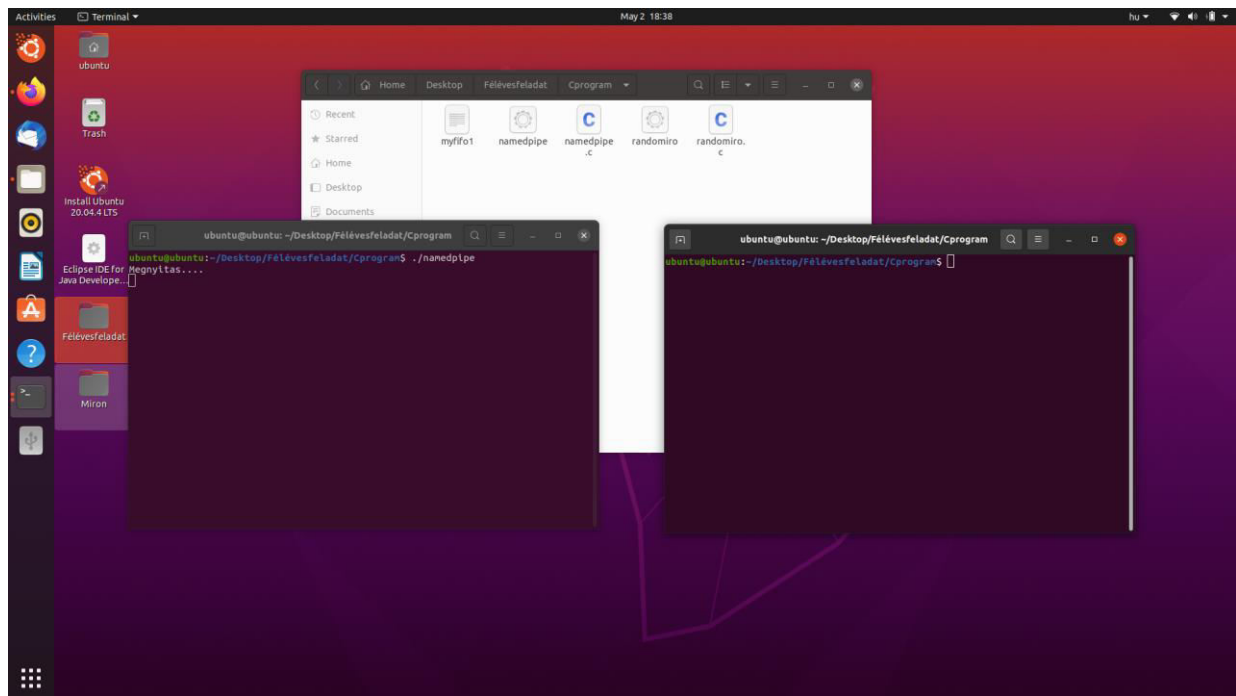


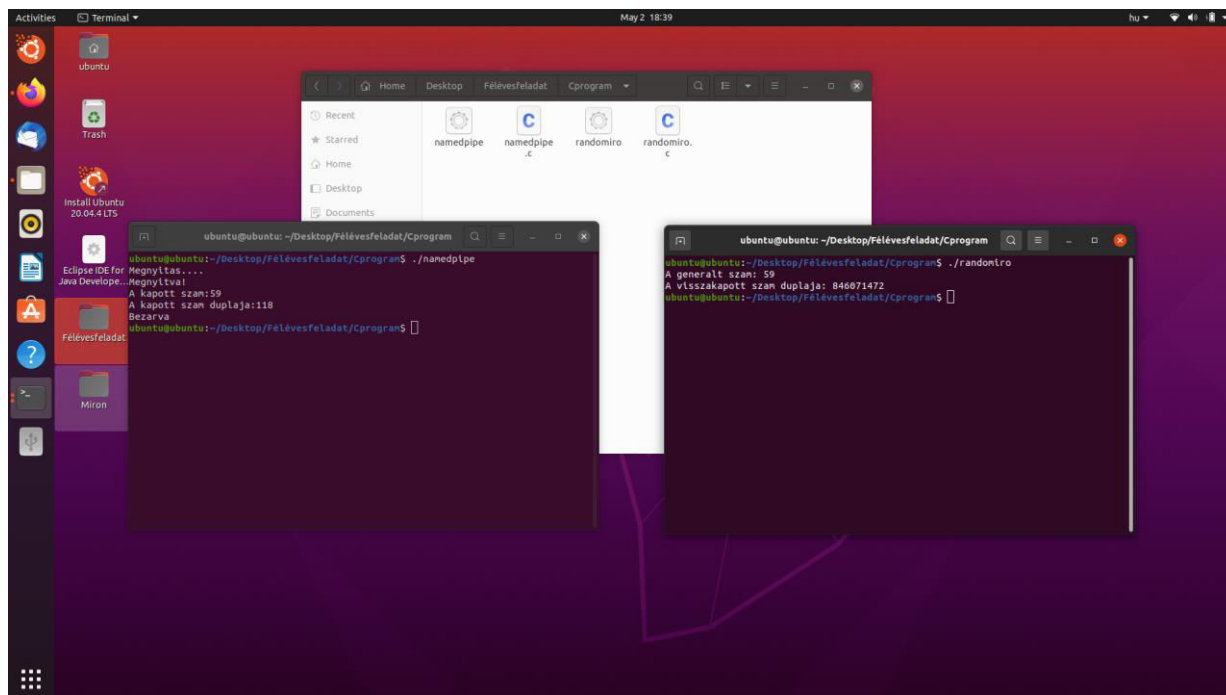
```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4
5 #include <sys/stat.h>
6 #include <sys/types.h>
7 #include <errno.h>
8 #include <fcntl.h>
9
10 int main (int argc, char* argv[])
11 {
12     //létrehozza a nevesített csövezeteket
13     if(mkfifo("myfifo", 0777) == -1) {
14         if(errno != EEXIST) {
15             printf("Could not create fifo file\n");
16             return 1;
17         }
18     }
19
20     //megnyitja
21     printf("Megnyitva...\n");
22     int fd = open("myfifo", O_RDONLY);
23     printf("Megnyitva!\n");
24
25     int randomNumber;
26
27     //kiolvassa a kapott számot és kiírja
28     read(fd, &randomNumber, sizeof(int));
29     printf("A kapott szám: %d\n", randomNumber);
30
31     //visszaküldi a dupláját
32     int dupla = randomNumber * 2;
33
34     printf("A kapott szám duplája: %d\n", dupla);
35     write(fd, &dupla, sizeof(int));
36
37
38     close(fd);
39     printf("Bezárva!\n");
40
41     return 0;
42 }
43
44
45
46 }
```



```
1#include <stdio.h>
2#include <stdlib.h>
3#include <unistd.h>
4
5#include <sys/stat.h>
6#include <sys/types.h>
7#include <errno.h>
8#include <fcntl.h>
9#include <time.h>
10
11int main (int argc, char* argv[])
12{
13    //legeneráljuk a random számot
14    srand(time(NULL));
15    int randomNumber = rand() % 100;
16
17    //megnyitjuk írásra a csővezeték
18    int fd = open("myfifo1", O_WRONLY);
19    if(fd == -1) {
20        return -1;
21    }
22    //beírjuk a számunkat
23    write(fd, &randomNumber, sizeof(int));
24    printf("A generált szám: %d\n", randomNumber);
25
26    //költsz a kapott számot és kiírja
27    int dupla;
28    read(fd, &dupla, sizeof(int));
29    printf("A visszakapott szám duplaja: %d\n", dupla);
30
31    //töröl a csővezeték
32    unlink("myfifo1");
33    close(fd);
34
35    return 0;
36}
```

A futtatás eredménye:





## 2. A feladat leírása:

6. Adott egy *igény szerinti lapozást* használó számítógéprendszer, melyben futás közben egy processz számára a következő laphivatkozással lehet hivatkozni: 6, 5, 4, 3, 5, 6, 2, 8, 5, 6, 5, 4, 7, 8, 4, 5, 6, 5, 5, 8

Memóriakeret (igényelt lapok): 3, ill. 4 memóriakeret.

Készítse el a laphivatkozások betöltését külön-külön táblázatba 3, ill. 4 memóriakeret esetén.

Mennyi laphiba keletkezik az alábbi algoritmusok esetén: FIFO, OPT?

Hasonlítsa össze és magyarázza az eredményeket!

---

A feladat elkészítésének lépései:

Automatikus mentés

Lapozási algoritmus

Keresés (Alt+Q)

mironkaa775@icloud.com

Fájl

Kezdőlap

Beszúrás

Lapelrendezés

Képletek

Adatok

Véleményezés

Nézet

Súgó

Beillesztés

Calibri

11

A

F

D

A

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

□

A futtatás eredménye:

Automatikus mentés

Lapozási algoritmus

Keresés (Alt+Q)

mironkaa775@icloud.com

Fájl

Kezdőlap

Beszúrás

Lapelrendezés

Képletek

Adatok

Véleményezés

Nézet

Súgó

Beillesztés

Formázás

Formázás táblázatként

Cellatiltások

Beszúrás

Törés

Formátum

Rendezés és szűrés

Keresés és kijelölés

Visszatérés

Válgólap

Betűtípus

Igazítás

Stílus

Cellák

Szerkesztés

M20

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

AA

AB

AC

AD

AE

AF

AG

AH

AI

AJ

AK

AL

AM

AN

AO

AP

AQ

AR

AS

AT

AU

1

FIFO

Laphivatkozások

FIFO

Laphivatkozások

2

Memóriakeret

6

5

4

3

5

6

2

8

5

6

5

4

7

8

4

5

6

5

5

8

Memóriakeret

6

5

4

3

5

6

2

8

5

6

5

4

7

8

4

5

6

5

5

8

Az OPT a legjobb lapcsere algoritmus a laphibákat tekintve, de a valóéletben nagyon nehéz használni, mert tudunk kéne a jövőbeli elemeket. Ezt figyelembe véve a FIFO 4 memóriakerettel jár a legkevesebb laphiba, amit hasznosítani is tudunk.