

Operációs rendszerek BSc

9. Gyak.

2022. 04. 05.

Készítette:

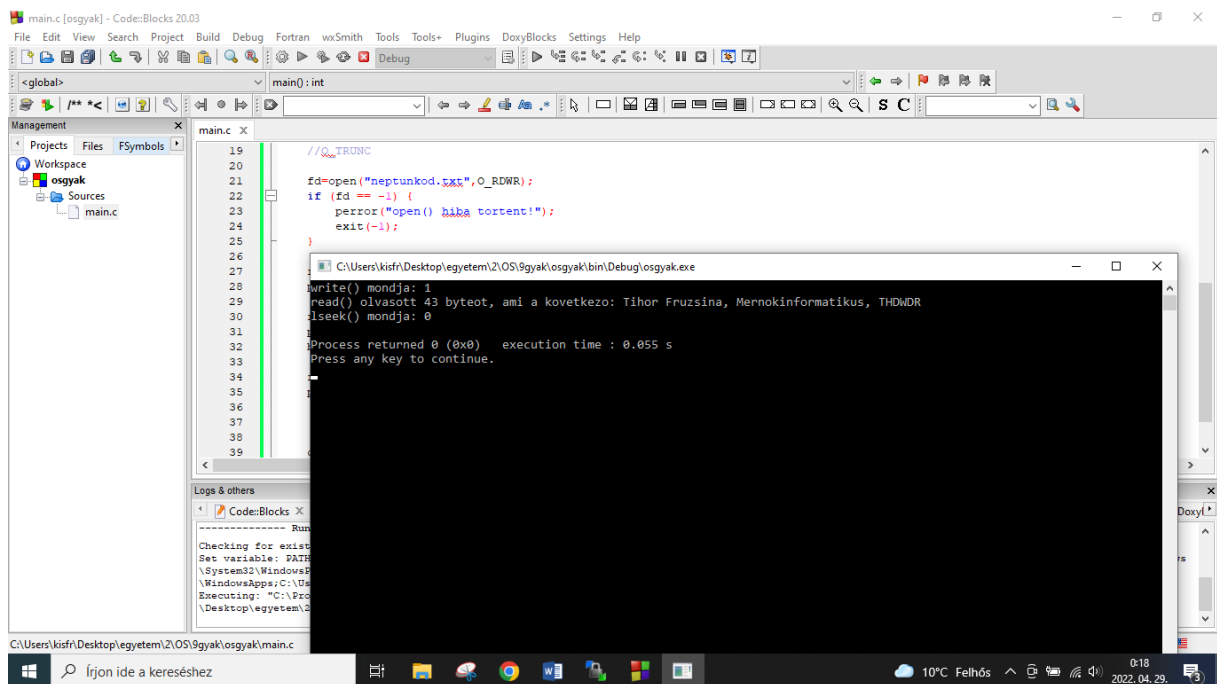
Tihor Fruzsina Bsc
Mérnökinformatikus
THDWDR

Miskolc, 2022

1.feladat: A tanult rendszerhívásokkal (open(), read()/write(), close()) - ők fogják a rendszerhívásokat tovább hívni - írjanak egy neptunkod_openclose.c programot, amely megnyit egy fájlt – neptunkod.txt, tartalma: hallgató neve, szak , neptunkod.

A program következő műveleteket végezze:

- olvassa be a neptunkod.txt fájlt, melynek attribútuma: O_RDWR
- hiba ellenőrzést,
- write() - mennyit ír ki a konzolra.
- read() - kiolvassa a neptunkod.txt tartalmát és mennyit olvasott ki (byte), és kiírja konzolra.
- lseek() – pozícionálja a fájl kurzor helyét, ez legyen a fájl eleje: SEEK_SET, és kiírja a konzolra.



```
19 //Q_0_TRUNC
20
21 fd=open("neptunkod.txt",O_RDWR);
22 if (fd == -1) {
23     perror("open() hiba történt!");
24     exit(-1);
25 }
26
27
28 write() mondja: 1
29 read() olvasott 43 byteot, ami a kovetkezo: Tihor Fruzsina, Mernokinformaticus, THDWDR
30 lseek() mondja: 0
31
32 Process returned 0 (0x0)   execution time : 0.055 s
33 Press any key to continue.
```

```
tihor@jerry:~/Documents/Gyakorlat/9$ ./vmi
write() mondja: 1
read() olvasott 42 byteot, ami a kovetkezo: Tihor Fruzsina, Mernokinformaticus, THDWDR
lseek() mondja: 0
tihor@jerry:~/Documents/Gyakorlat/9$
```

2. Készítse el a következő feladatot, melyben egy szignálkezelő több szignált is tud kezelni:

- a.) Készítsen egy szignál kezelőt (handleSignals), amely a SIGINT (CTRL + C) vagy SIGQUIT (CTRL + \) jelek fogására vagy kezelésére képes.
- b.) Ha a felhasználó SIGQUIT jelet generál (akár kill paranccsal, akár billentyűzetről a CTRL + \) a kezelő egyszerűen kiírja az üzenetet visszatérési értékét – a konzolra.
- c.) Ha a felhasználó először generálja a SIGINT jelet (akár kill paranccsal, akár billentyűzetről a CTRL + C), akkor a jelet úgy módosítja, hogy a következő alkalommal alapértelmezett műveletet hajtson végre (a SIG_DFL) – kiírás a konzolra.
- d.) Ha a felhasználó másodszor generálja a SIGINT jelet, akkor végrehajt egy alapértelmezett műveletet, amely a program befejezése - kiírás a konzolra.

Mentés: neptunkod_tobbszignal.c

```
^[\<Inside handler function for SIGALRM,db= 0
2 : Inside main function
Inside handler function for SIGALRM,db= 0
3 : Inside main function
^[iInside handler function for SIGALRM,db= 0
4 : Inside main function
^\
SIGQUIT: 3
5 : Inside main function
Inside handler function for SIGALRM,db= 0
6 : Inside main function
Inside handler function for SIGALRM,db= 0
7 : Inside main function
^\
SIGQUIT: 3
8 : Inside main function
Inside handler function for SIGALRM,db= 0
9 : Inside main function
Inside handler function for SIGALRM,db= 0
10 : Inside main function
^C
Var 5 masodpercet...
11 : Inside main function
Inside handler function for SIGALRM,db= 1
12 : Inside main function
Inside handler function for SIGALRM,db= 1
13 : Inside main function
^C
End of the program,exiting
```

3. Adott a következő ütemezési feladat, amit a FCFS, SJF és Round Robin (RR: 4 ms) ütemezési algoritmus alapján határozza meg következő **teljesítmény értékeket, metrikákat** (külön-külön táblázatba):

FCFS	P1	P2	P3	P4
Érkezés	0	0	2	5
CPU idő	24	3	6	3
Indulás	0	24	27	33
Befejezés	24	27	33	36
Várakozás	0	24	25	31

SJF	P1	P2	P3	P4
Érkezés	0	0	2	5
CPU idő	24	3	6	3
Indulás	12	0	3	9
Befejezés	36	3	9	12
Várakozás	12	0	1	4

RR(4 ms)	P1						P2	P3		P4
Érkezés	0	4	15	24	28	32	0	2	11	5
CPU idő	24	20	16	12	8	4	3	6	2	3
Indulás	0	11	20	24	28	32	4	7	18	15
Befejezés	4	15	24	28	32	36	7	11	20	18
Várakozás	0	7	5	0	0	0	4	5	7	10

Külön táblázatba számolja a teljesítmény értékeket!

CPU kihasználtság: számolni kell a **cs: 0,1(ms)** és **sch: 0,1 (ms)** értékkel is.

Algoritmus neve		FCFS
CPU kihasználtság		98,9%
Körülfordulási idők átlaga		28,3
Várakozási idők átlaga		20
Válaszidők átlaga		20

Algoritmus neve		SJF
CPU kihasználtság		98,9%
Körülfordulási idők átlaga		13,3
Várakozási idők átlaga		4,3
Válaszidők átlaga		4,3

Algoritmus neve		RR
CPU kihasználtság		95,8%
Körülfordulási idők átlaga		18,5
Várakozási idők átlaga		9,5
Válaszidők átlaga		4,8

Gyakorló feladatok - szignálkezelés

2. Írjon C nyelvű programot, amelyik kill() seg.-vel SIGALRM-et küld egy argumentumként megadott PID-u processznek, egy másik futó program a SIGALRM-hez rendeljen egy fv.-t amely kiírja pl. neptunkodot, továbbá pause() fv.-el blokkolódjon, majd kibillenés után jelezze, hogy kibillent és terminálódjon.

Mentés. neptunkod_gyak9_1.c

```
tihor@jerry:~/Documents/Gyakorlat/9/3$ gcc -o THDWR_gyak9_1 THDWR_gyak9_1.c
tihor@jerry:~/Documents/Gyakorlat/9/3$ ./THDWR_gyak9_1
1 : Inside main function
Tihor Fruzsina, THDWR
Kibillent!
2 : Inside main function
Tihor Fruzsina, THDWR
Kibillent!
3 : Inside main function
Tihor Fruzsina, THDWR
Kibillent!
4 : Inside main function
Tihor Fruzsina, THDWR
Kibillent!
tihor@jerry:~/Documents/Gyakorlat/9/3$
```

3. Írjon C nyelvű programot, amelyik a SIGTERM-hez hozzárendel egy fv.-t., amelyik kiírja az int paraméter értéket, majd végtelen ciklusban fusson, 3 sec-ig állandóan blokkolódva elindítás után egy másik shell-ben kill paranccsal (SIGTERM) próbálja terminálni, majd SIGKILL-el.”

Mentés. neptunkod_gyak9_2.c

```
tihor@jerry:~/Documents/Gyakorlat/9/4$ ./THDWR_gyak9_2
1 : Inside main function
Inside handler function for SIGTERM
2 : Inside main function
Inside handler function for SIGTERM
3 : Inside main function
Inside handler function for SIGTERM
4 : Inside main function
Inside handler function for SIGTERM
5 : Inside main function
```