Operációs rendszerek BSc

9. Gyak.

2022. 04. 04.

Készítette: Nagy Bence
Neptunkód: WH8L7E

Miskolc, 2022

1.feladat:

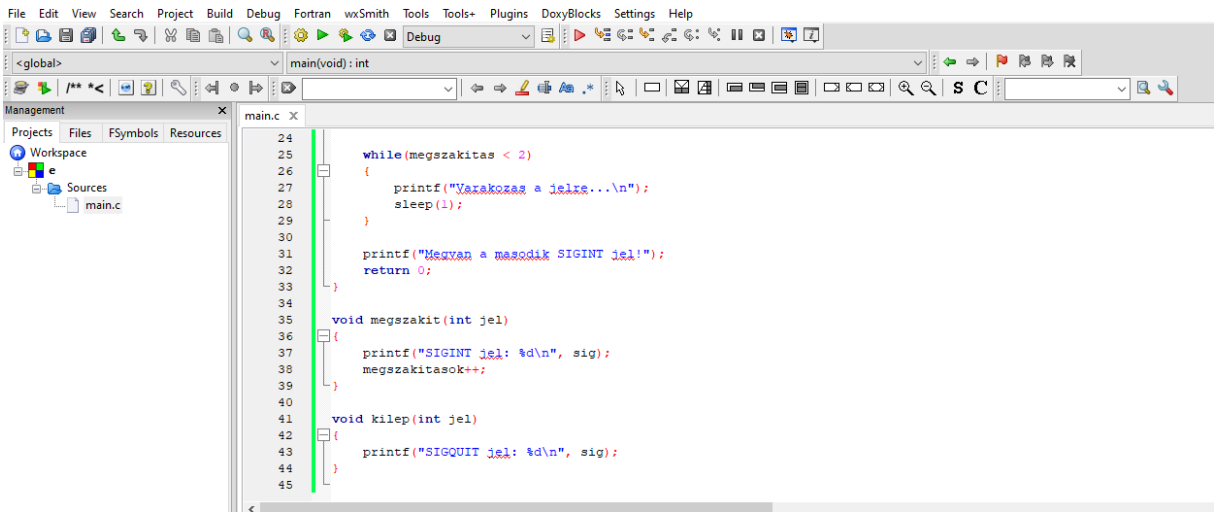
A tanult rendszerhívásokkal `(open(), read()/write(), close()` - ők fogják a rendszerhívásokat tovább hívni - írjanak egy `neptunkod_openclose.c` programot, amely megnyit egy fájlt – `neptunkod.txt`, tartalma: hallgató neve, szak , neptunkod
A program következő műveleteket végezze:

- 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.

```
nagyl40@jerry:~$ ./a.out
beolvasott tartalom: "Nagy Bence
GEIK
WH8L7E" osszesen "22" byte.
A mutato a fajl elejere lett allitva
A fajlba irtuk a(z) "Rendszerhivassal iras fajlba" szoveget. osszesen "29" byte.
```

2.feladat:

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.



3.feladat:

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):

Külön táblázatba számolja a teljesítmény értékeket!

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

FCFS	P1	P2	P3	P4		cpu kihasználtság:	0,988
érkezés	0	0	2	5		körülfordulási idők átlaga:	28,25
cpu idő	24	3	6	3		várakozási idők átlaga:	19,25
indulás	0	24	27	33		válaszidők átlaga:	19,25
befejezés	24	27	33	36			
várakozás	0	24	25	28			

SJF	P1	P2	P3	P4		cpu kihasználtság:	0,988
érkezés	0	0	2	5		körülfordulási idők átlaga:	23
cpu idő	24	3	6	3		várakozási idők átlaga:	14
indulás	3	0	27	33		válaszidők átlaga:	14
befejezés	27	3	33	36			
várakozás	3	0	25	28			

RR(4ms)	érkezés	cpu idő	indulás	befejezés	várakozás		cpu kihasználtság:	0,972
P1	0	24	0	4	0		körülfordulási idők átlaga:	18,75
P1*	4	20	7	11	3		várakozási idők átlaga:	5,625
P1**	11	16	15	19	4		válaszidők átlaga:	7,25
P1***	19	12	24	36	5			
P2	0	3	4	7	4			
P3	2	6	11	15	9			
P3*	15	2	19	21	4			
P4	5	3	21	24	16			

Gyakorló feladatok – szignálkezelés:

2. feladat:

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.

```
#define SECOND 1

void do_nothing();
void do_int();

main ()
{
    int i;
    unsigned sec=1;
    signal(SIGINT, do_int);

for (i=1;i<8;i++) {
    alarm(sec);
    signal(SIGALRM, do_nothing);
    printf("  %d varok...\n",i);
    pause();
    }
}

void do_nothing(){ ;}

void do_int() {
```

3.feladat:

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.”