Operációs rendszerek BSc

9.Gyak

2022.04.04.

Készítette:

Bodnár László BSc

Szak:
Mérnökinformatikus

Neptunkód: D1H8VP **2022.04.04.**

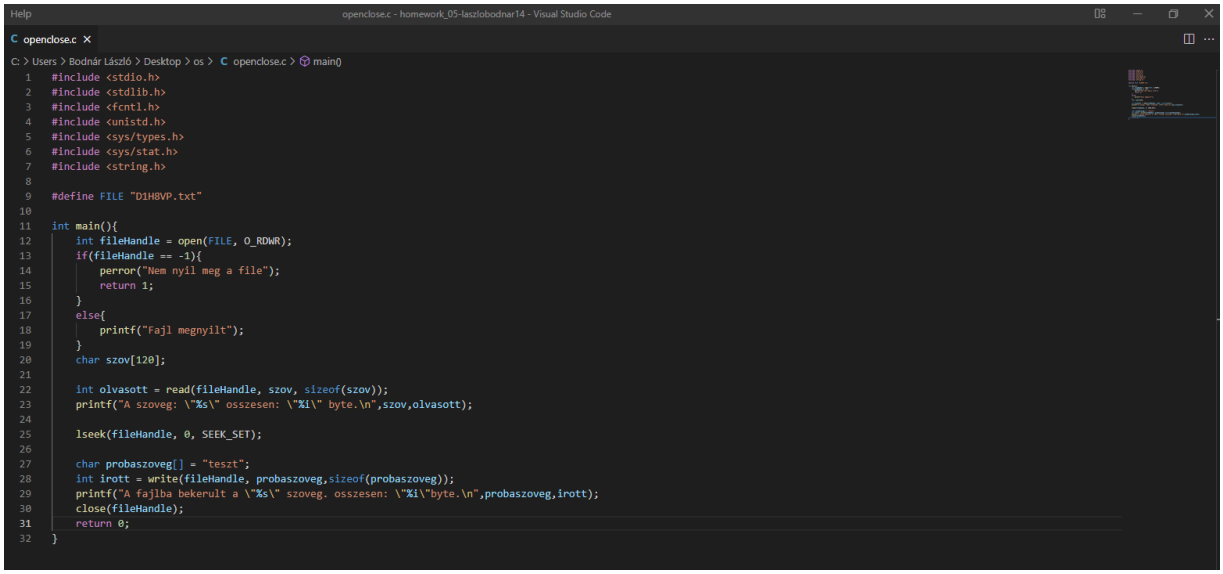
Feladatok

1. 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.



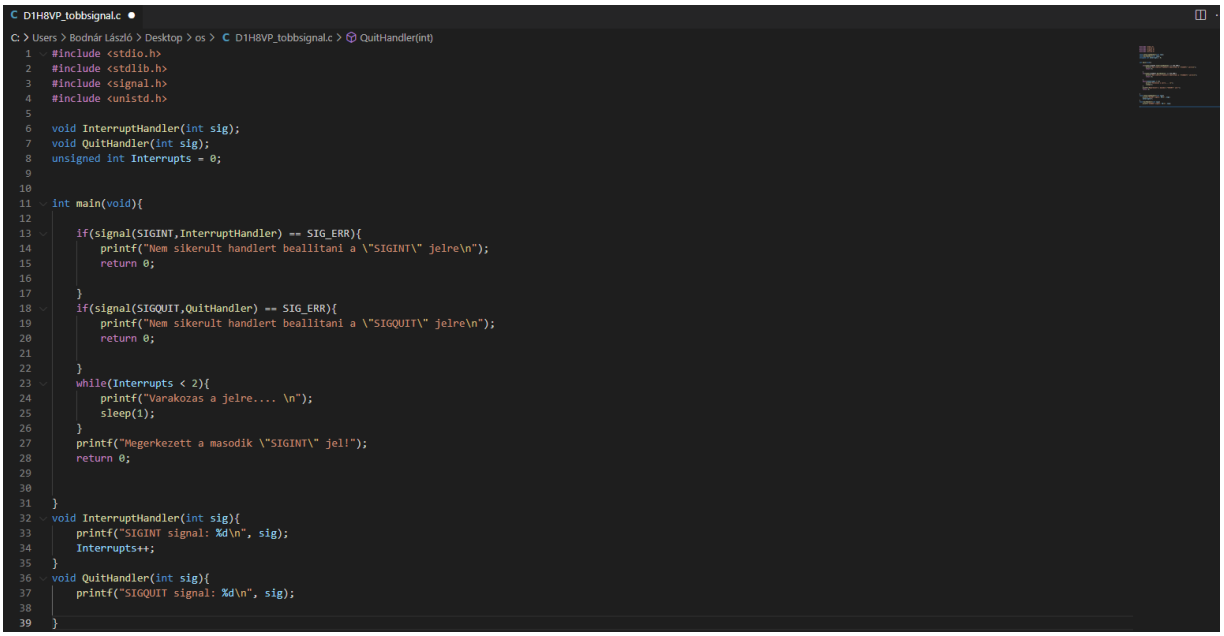
2. Készítse el a következő feladatot, melyben egy szignálkezelő több szignált is tud kezelni:

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.

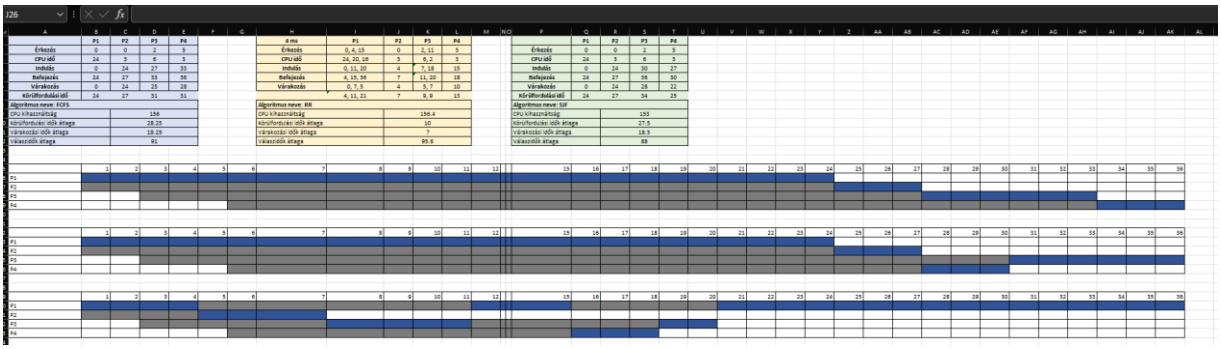
c.) Ha a felhasználó először generálja a SIGINT jelet (akár kill paranccsal, akár billentyűzetről a CTRL + C), akkor a jelet úgy módosítja, hogy a következő alkalommal alapértelmezett műveletet hajtson végre (a SIG_DFL) – kiírás a konzolra d.) Ha a felhasználó másodszor generálja a SIGINT jelet, akkor végrehajt egy alapértelmezett műveletet, amely a program

befejezése - kiírás a konzolra. Mentés: neptunkod_tobbszignal.c



```c
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>

void InterruptHandler(int sig);
void QuitHandler(int sig);
unsigned int Interrupts = 0;


int main(void){

    if(signal(SIGINT,InterruptHandler) == SIG_ERR){
        printf("Nem sikerult handlert beallitani a \"SIGINT\" jelre\n");
        return 0;

    }
    if(signal(SIGQUIT,QuitHandler) == SIG_ERR){
        printf("Nem sikerult handlert beallitani a \"SIGQUIT\" jelre\n");
        return 0;

    }
    while(Interrupts < 2){
        printf("Varakozas a jelre.... \n");
        sleep(1);
    }
    printf("Megerkezett a masodik \"SIGINT\" jel!");
    return 0;


}
void InterruptHandler(int sig){
    printf("SIGINT signal: %d\n", sig);
    Interrupts++;
}
void QuitHandler(int sig){
    printf("SIGQUIT signal: %d\n", sig);

}
```

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):



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

D1H8VP_gyak9_1.c

C: > Users > Bodnár László > Desktop > os > C D1H8VP_gyak9_1.c > kezelo(int)

```c
#include <stdio.h>
#include <sys/types.h>
#include <signal.h>
#include <unistd.h>

void kezelo(int i){
    printf("Signal kezeles: %d\n", i);
    return;
}
int main(void){

    printf("PID = %d\n", getpid());
    printf("Signal kezelo : %d\n", signal(SIGTERM,&kezelo));
    while(1){
        printf("lepes \n");
        sleep(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

D1H8VP_gyak9_1.c

C: > Users > Bodnár László > Desktop > os > C D1H8VP_gyak9_1.c > kezelo(int)

```c
#include <stdio.h>
#include <sys/types.h>
#include <signal.h>
#include <unistd.h>

void kezelo(int i){
    printf("Signal kezeles: %d\n", i);
    return;
}
int main(void){

    printf("PID = %d\n", getpid());
    printf("Signal kezelo : %d\n", signal(SIGTERM,&kezelo));
    while(1){
        printf("lepes \n");
        sleep(3);

    }

}
```