Operációs rendszerek BSc

9. Gyak.

2022. 04. 05.

Készítette:

Sziráczki Soma
Bsc

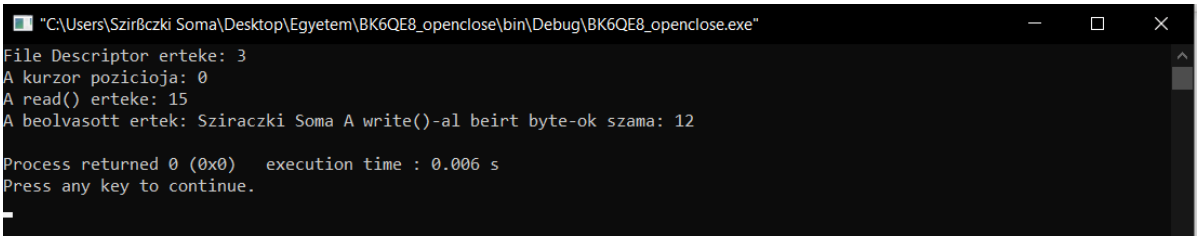
Programtervező
informatikus
BK6QE8

Miskolc, 2022

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.**

Megvalósítás:

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>

int main()
{
    char buf[20];
    int bufLenght;
    int fileDescriptor;
    int writeInfo;
    int readInfo;
    int seekInfo;

    //Fájl megnyitása

    fileDescriptor = open("BK6QEH.txt",O_RDWR);
    if (fileDescriptor == -1) {
        perror("open() hiba:");
        exit(fileDescriptor);
    }
    printf("File Descriptor értéke: %d\n",fileDescriptor);

    seekInfo = lseek(fileDescriptor, 0, SEEK_SET);
    if (seekInfo == -1){
        perror ("A pozicionalas nem volt sikeres.");
        exit(seekInfo);
    }
    printf("A kurzor pozíciója: %d\n", seekInfo);

    readInfo = read(fileDescriptor, buf, 15);
    if (readInfo == -1){
        perror("Az olvasas nem volt sikeres.");
        exit(seekInfo);
    }
    printf("A read() értéke: %d\n",readInfo);
    printf("A beolvasott ertek: %s", buf);

    strcpy (buf, "Ez egy teszt");
    bufLenght = strlen(buf);
    writeInfo = write(fileDescriptor, buf, bufLenght);
    if (writeInfo == -1){
        perror("Az iras nem volt sikeres.");
        exit(writeInfo);
    }
    printf("A write()-al beirt byte-ok szama: %d\n", writeInfo);
    return 0;
}
```



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

Megvalósítás:

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

void handleSignals(int signum);

int main() {
    void(*sigHandlerInterrupt)(int);
    void(*sigHandlerQuit)(int);
    void(*sigHandlerReturn)(int);
    sigHandlerInterrupt = sigHandlerQuit = handleSignals;
    sigHandlerReturn = signal(SIGINT, sigHandlerInterrupt);

    if (sigHandlerReturn == SIG_ERR) {
        perror("Signal error");
        return 1;
    }

    sigHandlerReturn = signal(SIGQUIT, sigHandlerQuit);

    if (sigHandlerReturn == SIG_ERR) {
        perror("Signal error");
        return 1;
    }
    for(;;) {
        printf("\nA programbol valo kilepeshez az alabbiakat kell elvegezni: \n");
        printf("1. masik terminal nyitasa.\n");
        printf("2. Adja ki a parancsot: kill: %d \n", getpid());
        sleep(10);
    }
    return 0;
}

void handleSignals(int signum) {

    if(signum == SIGINT)
        {
            printf("\n CTRL+C-t eszlelt\n");
            signal(SIGINT, SIG_DFL);
        }else if(signum == SIGQUIT)
            {
                printf("SIQUIT aktivalodott\n");
            }else
            {

                printf("\nFogadott jel szama: %d\n", signum);
            }

    return ;
}
```

```
sziraczki@sziraczki-VirtualBox:~/Asztal/forras2$ ./BK6QE8tobbszignal.o

A programbol valo kilepeshez az alabbiakat kell elvegezni:
1. masik terminal nyitasa.
2. Adja ki a parancsot: kill: 9162
```

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

Megvalósítás:

FCFS	P1	P2	P3	P4	sum
Érkezés	0	0	2	5	
CPU idő	24	3	6	3	36
Indulás	0	24	27	33	
Befejezés	24	27	33	36	
Várakozás	0	24	25	28	
Körülfordu	24	27	31	31	

CPU Kihasználtság	88.89%
Körülfordlási idők átlaga	28.25
Várakozási idők átlaga	19.25
Válaszidől átlaga	19.25

SJF	P1	P2	P3	P4	sum
Érkezés	0	0	2	5	
CPU idő	24	3	6	3	36
Indulás	12	0	3	9	
Befejezés	36	3	9	12	
Várakozás	12	0	1	4	
Körülfordu	36	3	7	9	

CPU Kihasználtság	88.89%
Körülfordlási idők átlaga	13.75
Várakozási idők átlaga	4.25
Válaszidől átlaga	4.25

RR: 4ms	P1	P2	P3	P4	sum
Érkezés	, 15, 24, 28	0	2, 11	5	
CPU idő	20, 16, 12,	3	6, 2	3	98
Indulás	1, 20, 24, 28	4	7, 18	15	
Befejezés	5, 24, 28, 32	7	11, 20	18	
Várakozás	, 7, 5, 0, 0,	4	5, 7	10	
Körülfordu	96	9	20	13	

CPU Kihasználtság	89.80%
Körülfordlási idők átlaga	34.5
Várakozási idők átlaga	9.5
Válaszidől átlaga	9.5