Operációs rendszerek BSc

9. Gyak.

2022. 04. 05.

Készítette:

Sziráczki Soma
Bsc

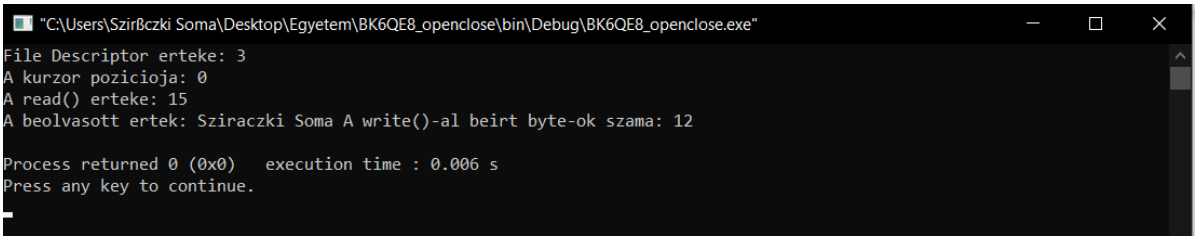
Programtervező
informatikus
BK6QE8

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("BK6QE8.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 értek: %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 száma: %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: