Operációs Rendszerek BSc

9. Gyak.

2022.04.05.

Készítette:

Tóth Dániel Márk BSc

Mérnökinformatika

IEFMWZ

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.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <signal.h>
#pragma warning(disable:4996)

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

	int i;
	FILE* fptr;
	char fn[50];
	char str[] = "Toth Daniel Mark, GEIK, IEFMWZ\n";
	fptr = fopen("IEFMWZ.txt", "w");
	for (i = 0; str[i] != '\n'; i++) {
		fputc(str[i], fptr);
	}
	fclose(fptr);

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

	seekInfo = lseek(fileDescriptor, 0, SEEK_SET);
	if (seekInfo == -1)
	{
		perror("lseek hiba:");
		exit(seekInfo);
	}
	printf("A kurzor pozicioja: %d\n", seekInfo);

	readInfo = read(fileDescriptor, buf, 15);
	if (seekInfo == -1)
	{
		perror("read hiba!");
		exit(readInfo);
	}
	printf("Olvasott byte-ok száma: %d", readInfo);

	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 beírt byte-ok szama: %d", writeInfo);

}
```

Ebben a kódban a feladat szerint előbb létrehoztam a fájlt, majd az O_RDWR műveletet elvégezve beolvastam, majd analizáltam a fájlt.

A feladat szerint pedig beleírtam a fájlba az előre megadott szavakat, majd elmentettem a fájlt.

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

26 lines (22 sloc) | 539 Bytes

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

void sigdfl_handler(int signum)
{
        printf("\nEZUTAN A DEFAULT SIGNAL METODUS FUT LE!\n");
        printf("Ami a program befejezese, nyomjon meg barmilyen gombot a kilepeshez!\n");
}

void sig_handler(int signum) {

        printf("\nBEIRTAD A SIGINT KILEPO KODJAT!\n");
        signal(SIG_DFL, sigdfl_handler);
}

int main()
{
        signal(SIGINT, sig_handler);
        for (int i = 1;; i++) {
                printf("%d : IRD BE CTRL+C\n", i);
        }
        return 0;
}
```

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	28

CPU Kihasználtság: 99.42%
Körülfordulási idők átlaga: 28.25
Várakozási idők átlaga: 19.25
Válaszidők átlaga: 19.25

SJF	P1	P2	P3	P4
Érkezés	0	0	2	5
CPU Idő	24	3	6	3
Indulás	0	24	27	33
Befejezés	36	3	9	12
Várakozás	12	0	1	4

CPU Kihasználtság: 99.54%

Körülfordulási idők átlaga: 13.25

Várakozási idők átlaga: 4.25

Válaszidők átlaga: 4.25

RR	P1	P2	P3	P4
Érkezés	0	0	2	5
CPU Idő	24	3	6	3
Indulás	0	24	27	33
Befejezés	36	7	20	18
Várakozás	12	4	12	10

CPU Kihasználtság: 99.6%

Körülfordulási idők átlaga: 18.5

Várakozási idők átlaga: 9.5

Válaszidők átlaga: 9.5