Operációs rendszerek BSc

1. Gyak.

2022. 04. 10.

Készítette:
Hauer Attila Árpád
Szak Mérnökinformatikus
Neptunkód JJL4WE

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 <fcntl.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <string.h>

#define FILE "JJL4WE.txt"

int main()
{
    int fileHandle = open(FILE, O_RDWR);
    if(fileHandle == -1)
    {
        perror("Nem sikerult megnyitni a fajlt!");
        return 1;
    }
    else
    {
        printf("Sikeres volt a fajl megnyitasa!\n");
    }
    char tartalom[120];

    int olvasott = read(fileHandle, tartalom, sizeof(tartalom));
    printf("Beolvasott tartalom: \"%s\" osszesen: \"%i\" byte.\n",tartalom,olvasott);

    lseek(fileHandle, 0, SEEK_SET);

    char text[] = "teszt";
    int irt = write(fileHandle, text, sizeof(text));
    printf("A fajlba irtuk a(z) \"%s\" szoveget. Osszesen: \"%i\" byte. \n", text, irt);

    close(fileHandle);

    return 0;
}
```

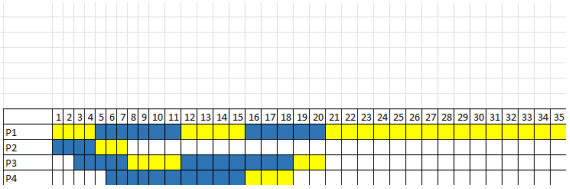
2. Készítse el a következő feladatot, melyben egy szignálkezelő több szignált is tud kezelni: a.) Készítsen egy szignál kezelőt (handleSignals), amely a SIGINT (CTRL + C) vagy SIGQUIT (CTRL + \) jelek fogására vagy kezelésére képes. b.) Ha a felhasználó SIGQUIT jelet generál (akár kill paranccsal, akár 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

```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(sigHandlerInterrupt == SIG_ERR)
    {
        perror("Signal error");
        return 1;

    }

    for(;;)
    {
        printf("A program leallitasahoz a kovetkezoket vegezze el: \n");
        printf("1. Nyisson meg egy masik terminalt.\n");
        printf("2. Adja ki a parancsot: kill: %d \n", getpid());
        sleep(10);
    }

    return 0;
}

void handleSignals(int signum)
{
    switch(signum)
    {
        case SIGINT:
            printf("\n CTRL+C-t észlelt\n");
            signal(SIGINT, SIG_DFL);
            break;
        case SIGQUIT:
            printf("SIQUIT aktivalodott\n");
            break;
        default:
            printf("\nFogadott jel szama: %d\n", signum);
            break;
    }
```

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):

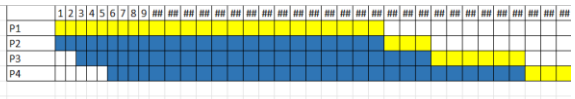
4 ms	P1	P2	P3	P4	
Érkezés	0, 4, 15	0	2, 11	5	P2, P3, P1
CPU idő	24, 20, 16	3	6, 2	3	P3, P1, P4
Indulás	0, 11, 20	4	7, 18	15	P1, P4, P3
Befejezés	4, 15, 36	7	11, 20	18	P4, P3, P1
Várakozás	0, 7, 5	4	5, 7	10	P3, P1
	4, 11, 21	7	9, 9	13	

Algoritmus neve: RR	
CPU kihasználtság	156,4
Körülfordulási idők átlaga	10
Várakozási idők átlaga	7
Válaszidők átlaga	95,6



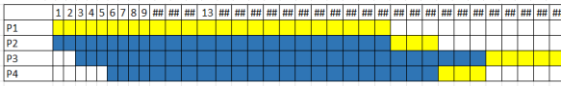
	P1	P2	P3	P4
Érkezés	0	0	2	5
CPU idő	24	3	6	3
Indulás	0	24	27	33
Befejezés	24	27	33	36
Várakozás	0	24	25	28
Körülfordulási idő	24	27	31	31

Algoritmus neve: FCFS	
CPU kihasználtság	156
Körülfordulási idők átlaga	28,25
Várakozási idők átlaga	19,25
Válaszidők átlaga	91



	P1	P2	P3	P4
Érkezés	0	0	2	5
CPU idő	24	3	6	3
Indulás	0	24	30	27
Befejezés	24	27	36	30
Várakozás	0	24	28	22
Körülfordulási idő	24	27	34	25

Algoritmus neve: SJF	
CPU kihasználtság	153
Körülfordulási idők átlaga	27,5
Várakozási idők átlaga	18,5
Válaszidők átlaga	88



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

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

void handleSigalarm();

int main()
{
    printf("A program pidje: %d\n",getpid());
    signal(SIGALRM, handleSigalarm);
    pause();
    printf("Kibillent\n");
    exit(0);
    return 0;
}

void handleSigalarm()
{
    printf("JJL4WE\n");
}
```

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

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

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

int main()
{
    printf("PID: %d\n",getpid());
    printf("Signal kezelo atvetele: %d\n",signal(SIGTERM,&kezelo));
    while(1)
    {
        printf("lep\n");
        sleep(3);
    }

    return 0;
}
```