

Operációs rendszerek BSc

9. Gyak.

2022. 04. 05.

Készítette:

Berecz Antónia Bsc

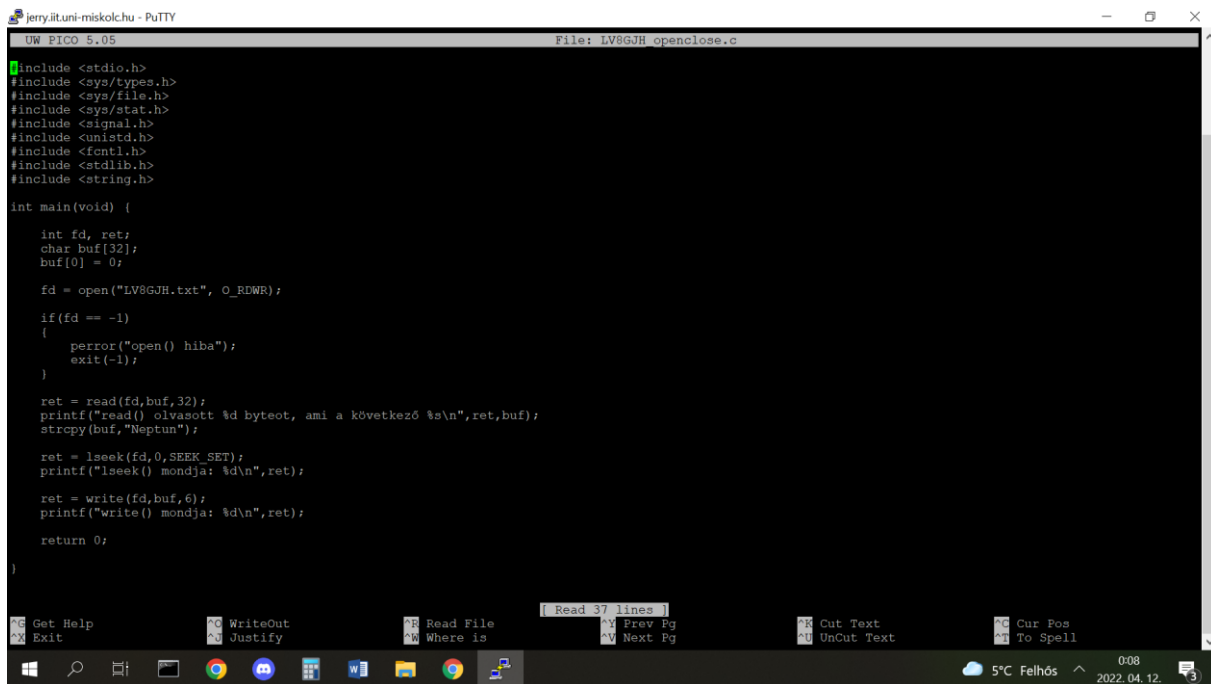
Programtervező informatikus

LV8GJH

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.



```
File: LV8GJH_openclose.c
#include <stdio.h>
#include <sys/types.h>
#include <sys/file.h>
#include <sys/stat.h>
#include <signal.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>

int main(void) {
    int fd, ret;
    char buf[32];
    buf[0] = 0;

    fd = open("LV8GJH.txt", O_RDWR);

    if(fd == -1)
    {
        perror("open() hiba");
        exit(-1);
    }

    ret = read(fd, buf, 32);
    printf("read() olvasott %d byteot, ami a következő %s\n", ret, buf);
    strcpy(buf, "Neptun");

    ret = lseek(fd, 0, SEEK_SET);
    printf("lseek() mondja: %d\n", ret);

    ret = write(fd, buf, 6);
    printf("write() mondja: %d\n", ret);

    return 0;
}
```

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

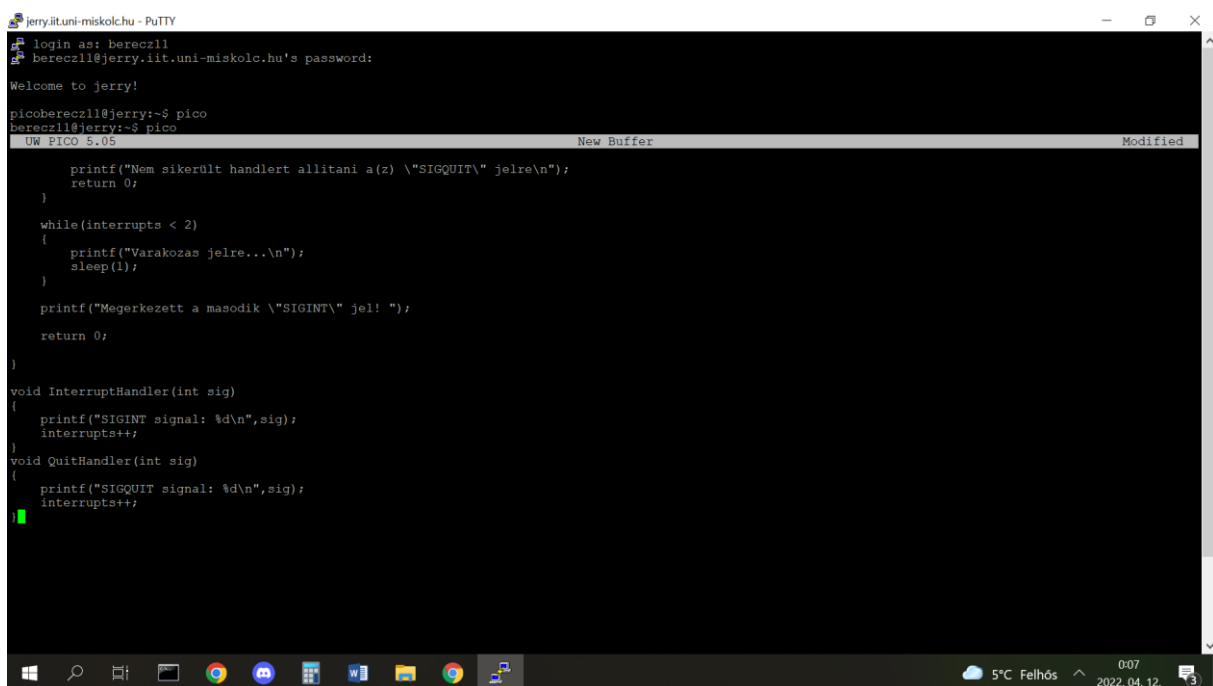
a.) Készítsen egy szignál kezelőt (handleSignals), amely a SIGINT (CTRL + C) vagy SIGQUIT (CTRL + \) jelek fogására vagy kezelésére képes.

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_tobbszinal.c



```
jerry.iit.uni-miskolc.hu - PuTTY
login as: berecz11
berez11@jerry.iit.uni-miskolc.hu's password:
Welcome to jerry!
picoberez11@jerry:~$ pico
berez11@jerry:~$ pico
UW PICO 5.05
New Buffer
Modified

    printf("Nem sikerült handlert allitani a(z) \"SIGQUIT\" jelre\n");
    return 0;
}

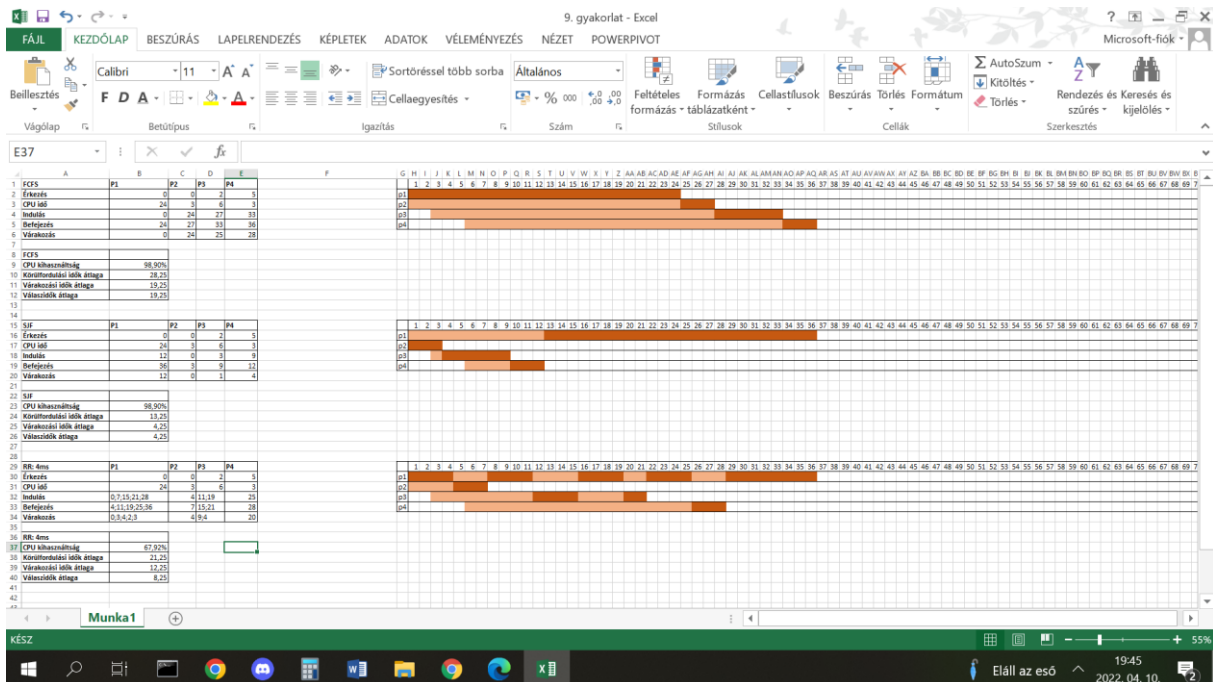
while(interrupts < 2)
{
    printf("Varakozas jelre...\n");
    sleep(1);
}

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);
    interrupts++;
}
```

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

```

jerry@it.uni-miskolc.hu - Putty
UW PICO 5.05
New Buffer
Modified

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

void do_nothing(int pid);

int main(void) {
    printf("PID = %d\n", getpid());
    signal(SIGTERM, do_nothing);
    printf("Varakozas\n");
    pause();
    printf("Megjott az alarm\n");
    return 0;
}

void do_nothing(int pid)
{
    printf("do_nothing() fut");
}

```

The screenshot shows a PuTTY terminal window titled "jerry.ituni-miskolc.hu - PuTTY". The terminal displays the following C code:

```
UW PICO 5.05 New Buffer Modified

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

int main(int argc, char **argv) {
    int pid;
    if(argc<1)
    {
        perror("Nincs kinek");
    }

    pid = atoi(argv[1]);
    kill(pid,SIGTERM);
}
```

The terminal window has a menu bar with options: Get Help, Exit, WriteOut, Justify, Read File, Where is, Prev Pg, Next Pg, Cut Text, UnCut Text, Cur Pos, To Spell. The system tray at the bottom shows the date and time as 2022.04.12. 0:13 and the weather as 5°C Tiszta égbolt.

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

The screenshot shows a PuTTY terminal window titled "jerry.ituni-miskolc.hu - PuTTY". The terminal displays the following C code:

```
UW PICO 5.05 New Buffer Modified

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

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

int main(void) {
    printf("PID =%d\n",getpid());
    printf("Signal kezelo atvetele: %d \n",signal(SIGTERM,kezelo));

    while(1)
    {
        printf("lepes\n");
        sleep(3);
    }
    return 0;
}
```

The terminal window has a menu bar with options: Get Help, Exit, WriteOut, Justify, Read File, Where is, Prev Pg, Next Pg, Cut Text, UnCut Text, Cur Pos, To Spell. The system tray at the bottom shows the date and time as 2022.04.12. 0:14 and the weather as 5°C Tiszta égbolt.

