# Operációs rendszerek BSc

9. Gyak.

2022. 04. 06.

**Készítette:**
Gyáni Kevin Zsolt Bsc
Szak Programtervező informatikus
Neptunkód CBOYZF

**Miskolc, 2022**

# 1.feladat

```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("CBOYZF.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;

}
```

Az open paranccsal megnyitottuk az adatfolyamot a fájl felé. O_RDWR beállítjuk hogy a fájlt írni és olvasni is lehessen. A read()-el olvassuk a fájlt. A write()-al pedig írjuk.

# 2.feladat

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


unsigned int interrupts = 0;
void InterruptHandler(int sig);
void QuitHandler(int sig);

int main(void) {

    if(signal(SIGINT,InterruptHandler) == SIG_ERR)
    {
        printf("Nem sikerült handlert allitani a(z) \"SIGINT\" jelre\n");
        return 0;
    }

    if(signal(SIGQUIT, QuitHandler) == SIG_ERR)
    {
        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++;
}
```

A SIGINT figyeli a ctr + c interrupt-ot, a signal pedig össze köti ezt az InterruptHandler() metódussal. A SIGQUIT figyeli a ctr + \ interrupt-ot, a signal pedig össze köti ezt az QuitHandler() metódussal.

## 3.feladat

## A megoldások az excel táblázatbna találhtóak!

# 4.feladat

```c
#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("Varok de meddig?\n");
    pause();
    printf("Vegre, itt az alarm\n");
    return 0;
}
void do_nothing(int pid)
{
    printf("do_nothing() fut");
}
```

```c
#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);
}
```

Az első program létrehoz egy processzt amit várakoztat, a második program megöli azt.

## 5.feladat

```
#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;
}
```

Folyamatosan futtatja a processzt kiírja a processz id-t 3 mp-ként megtesz egy „lépést”.