

# **Operációs rendszerek BSc**

**9. Gyak.**

**2022. 04. 04.**

**Készítette:**

Kórád György Bsc  
Programtervező Informatikus  
ZF440N

**Miskolc, 2022**

A program következő műveleteket végezze:

- ```
Terminál - gyuri@x54c:~/OS/ZF440N0sGyak/ZF440N_0404/Források/1-feladat
```
- ```
Fájl Szerkesztés Nézet Terminál Lapok Súgó
```
- ```
GNU nano 6.2 ZF440N_openclose.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 "ZF440N.txt"

int main() {
    int fileHandle = open(FILE, O_RDWR);
    if(fileHandle == -1)
    {
        perror("Nem sikerult megnyitni a fajlt!");
        return 1;
    } else
        printf("Megnyitottam a fajlt!");
    char tartalom[64];

    int olvasott = read(fileHandle, tartalom, sizeof(tartalom));
    printf("beolvasott tartalom: \"%s\" osszesen \"%i\" byte.\n", tartalom, olvaso>
```
- ```
lseek(fileHandle, 0, SEEK_SET);

char szoveg[] = "teszt";
int irt = write(fileHandle, szoveg, sizeof(szoveg));
printf("A fajlba irtuk a(z) \"%s\" szoveget. osszesen \"%i\" byte.\n", szoveg,>
```
- ```
close(fileHandle);
return 0;
}
```
- ```
or directory

osszesen "0" byte.
n "6" byte.
```
- ```
rád György, ZF440N, Programtervező Informatikus" osszesen "53" byte.
```
- ```
"6" byte.
```
- ```
G Heln Kíírás Keresés Cut Execute Location
K pész Beolvasás Csere U Paste Sorkizárás/ Ugrás sorra
```

```
Terminál - gyuri@x54c:~/OS/ZF440N0sGyak/ZF440N_0404/Forrasok/1-feladat
Fájl Szerkesztés Nézet Terminál Lapok Súgó
GNU nano 6.2 ZF440N.txt
Kórád György, ZF440N, Programtervező Informatikus
≡ ZF440N.txt
1 Kórád György, ZF440N, Programtervező Informatikus
U
U
U

[ 1 sor beolvasva ]
^G Help      ^O Kiírás    ^W Keresés   ^K Cut       ^T Execute   ^C Location
^X Kilépés   ^R Beolvasás ^\ Csere     ^U Paste     ^J Sorkizárás ^_ Ugrás sorra
```

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.

```
[gyuri@x54c 2-feladat]$ cat ZF440N_tobbszignal.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 (sigHandlerReturn == SIG_ERR) {
        perror("Signal error");
        return 1;
    }

    for(;;) {
        printf("\nA program leallitasahoz a kovetkezoeket 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 eszlelt\n");
            signal(SIGINT, SIG_DFL);
            break;
        case SIGQUIT:
            printf("SIGQUIT aktivalodott\n");
            break;
        default:
            printf("\nFogadott jel szama: %d\n", signum);
            break;
    }
    return ;
}
```

```
Terminál - gyuri@x54c: ~/OS/ZF440N0sGyak/ZF440N_0404/Forrasok/2-feladat
Fájl Szerkesztés Nézet Terminál Lapok Súgó

A program leállításához a következőket végezze el:
1. Nyisson meg egy másik terminált.
2. Adja ki a parancsot: kill: 9016

A program leállításához a következőket végezze el:
1. Nyisson meg egy másik terminált.
2. Adja ki a parancsot: kill: 9016

A program leállításához a következőket végezze el:
1. Nyisson meg egy másik terminált.
2. Adja ki a parancsot: kill: 9016

A program leállításához a következőket végezze el:
1. Nyisson meg egy másik terminált.
2. Adja ki a parancsot: kill: 9016
Befejeződött
[gyuri@x54c 2-feladat]$ █

Terminál - gyuri@x54c: ~/OS/ZF440N0sGyak/ZF440N_0404/Forrasok/2-feladat
Fájl Szerkesztés Nézet Terminál Lapok Súgó

[gyuri@x54c 2-feladat]$ kill 9016
[gyuri@x54c 2-feladat]$ █
```

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

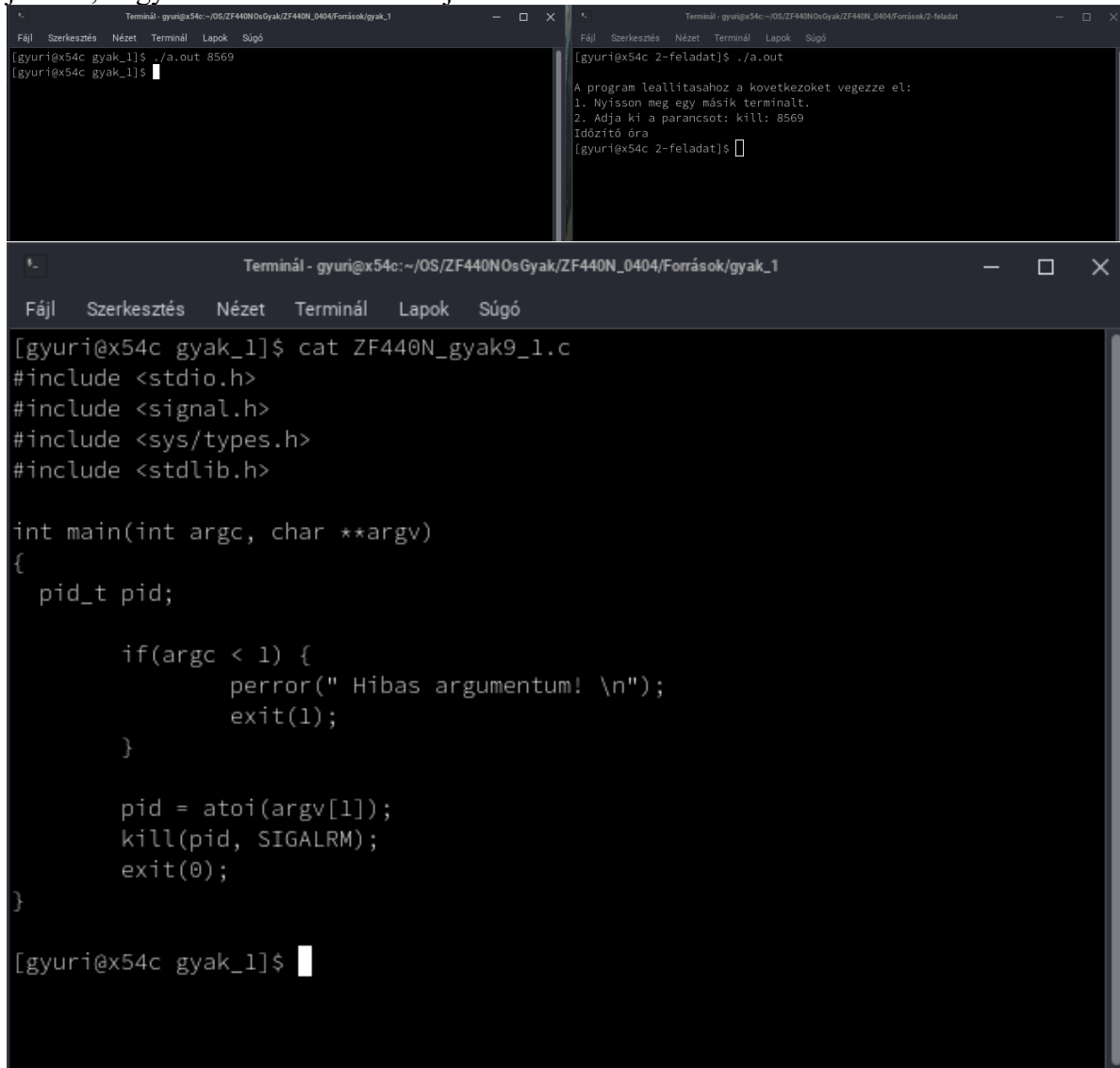
RR: 4ms	P1	P2	P3	P4			Round Robin [4ms]	
Érkezés	0, 4, 15, 24, 28, 32	0	2, 11	5	98		CPU Kihasználtság	89,80%
CPU idő	24, 20, 16, 12, 8, 4	3	6, 2	3			Körülfordlási idők átlaga	34,5
Indulás	0, 11, 20, 24, 28, 32	4	7, 18	15			Várakozási idők átlaga	9,5
Befejezés	4, 15, 24, 28, 32, 36	7	11, 20	18			Válaszidő átlaga	9,5
Várakozás	0, 7, 5, 0, 0, 0	4	5, 7	10				
Körülfordulási idő	96	9	20	13				

SJF	P1	P2	P3	P4			SJF	
Érkezés	0	0	2	5	36		CPU Kihasználtság	88,89%
CPU idő	24	3	6	3			Körülfordlási idők átlaga	13,75
Indulás	12	0	3	9			Várakozási idők átlaga	4,25
Befejezés	36	3	9	12			Válaszidő átlaga	4,25
Várakozás	12	0	1	4				
Körülfordulási idő	36	3	7	9				

FCFS	P1	P2	P3	P4	Össz		FCFS	
Érkezés	0	0	2	5	36		CPU Kihasználtság	88,89%
CPU idő	24	3	6	3			Körülfordlási idők átlaga	28,25
Indulás	0	24	27	33			Várakozási idők átlaga	19,25
Befejezés	24	27	33	36			Válaszidő átlaga	19,25
Várakozás	0	24	25	28				
Körülfordulási idő	24	27	31	31				

# Gyakorló feladatok – szignálkezelés

1. 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.



The image shows three terminal windows illustrating the execution of a C program for signal handling.

The top-left terminal window shows the command `./a.out 8569` being executed, resulting in the prompt `[gyuri@x54c gyak_1]$`.

The top-right terminal window shows the command `./a.out` being executed, displaying instructions for the program and the prompt `[gyuri@x54c 2-feladat]$`.

The bottom terminal window shows the source code of the program `ZF440N_gyak9_1.c`:

```
[gyuri@x54c gyak_1]$ cat ZF440N_gyak9_1.c
#include <stdio.h>
#include <signal.h>
#include <sys/types.h>
#include <stdlib.h>

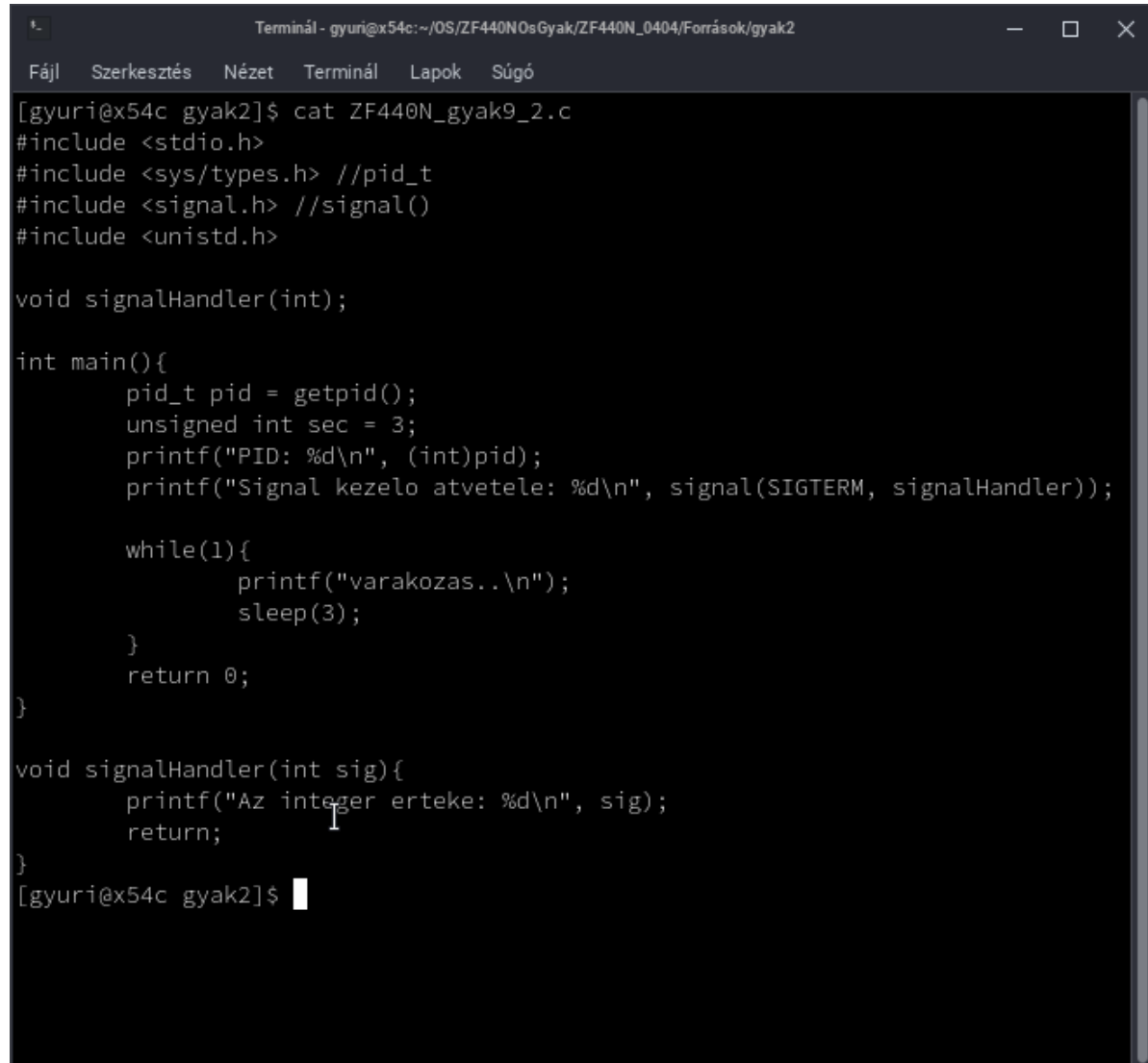
int main(int argc, char **argv)
{
    pid_t pid;

    if(argc < 1) {
        perror(" Hibas argumentum! \n");
        exit(1);
    }

    pid = atoi(argv[1]);
    kill(pid, SIGALRM);
    exit(0);
}
```

The bottom terminal window shows the prompt `[gyuri@x54c gyak_1]$`.

2. Írjon C nyelvű programot, amelyik a SIGTERM-hez hozzárendel egy fv-t., amelyik kiírja az int paraméter értékét, majd végtelen ciklusban fusson, 3 sec-ig állandóan blokkolódva elindítás után egy másik shell-ben kill paranccsal (SIGTERM) próbálja terminálni, majd SIGKILL-el.”



```
Terminál - gyuri@x54c: ~/OS/ZF440N0sGyak/ZF440N_0404/Forrasok/gyak2
Fájl Szerkesztés Nézet Terminál Lapok Súgó

[gyuri@x54c gyak2]$ cat ZF440N_gyak9_2.c
#include <stdio.h>
#include <sys/types.h> //pid_t
#include <signal.h> //signal()
#include <unistd.h>

void signalHandler(int);

int main(){
    pid_t pid = getpid();
    unsigned int sec = 3;
    printf("PID: %d\n", (int)pid);
    printf("Signal kezelo atvetele: %d\n", signal(SIGTERM, signalHandler));

    while(1){
        printf("varakozas..\n");
        sleep(3);
    }
    return 0;
}

void signalHandler(int sig){
    printf("Az integer erteke: %d\n", sig);
    return;
}

[gyuri@x54c gyak2]$
```