

# Operációs rendszerek BSc

9. Gyak.

2022. 04. 06.

**Készítette:**

Szelényi Szabolcs Bsc

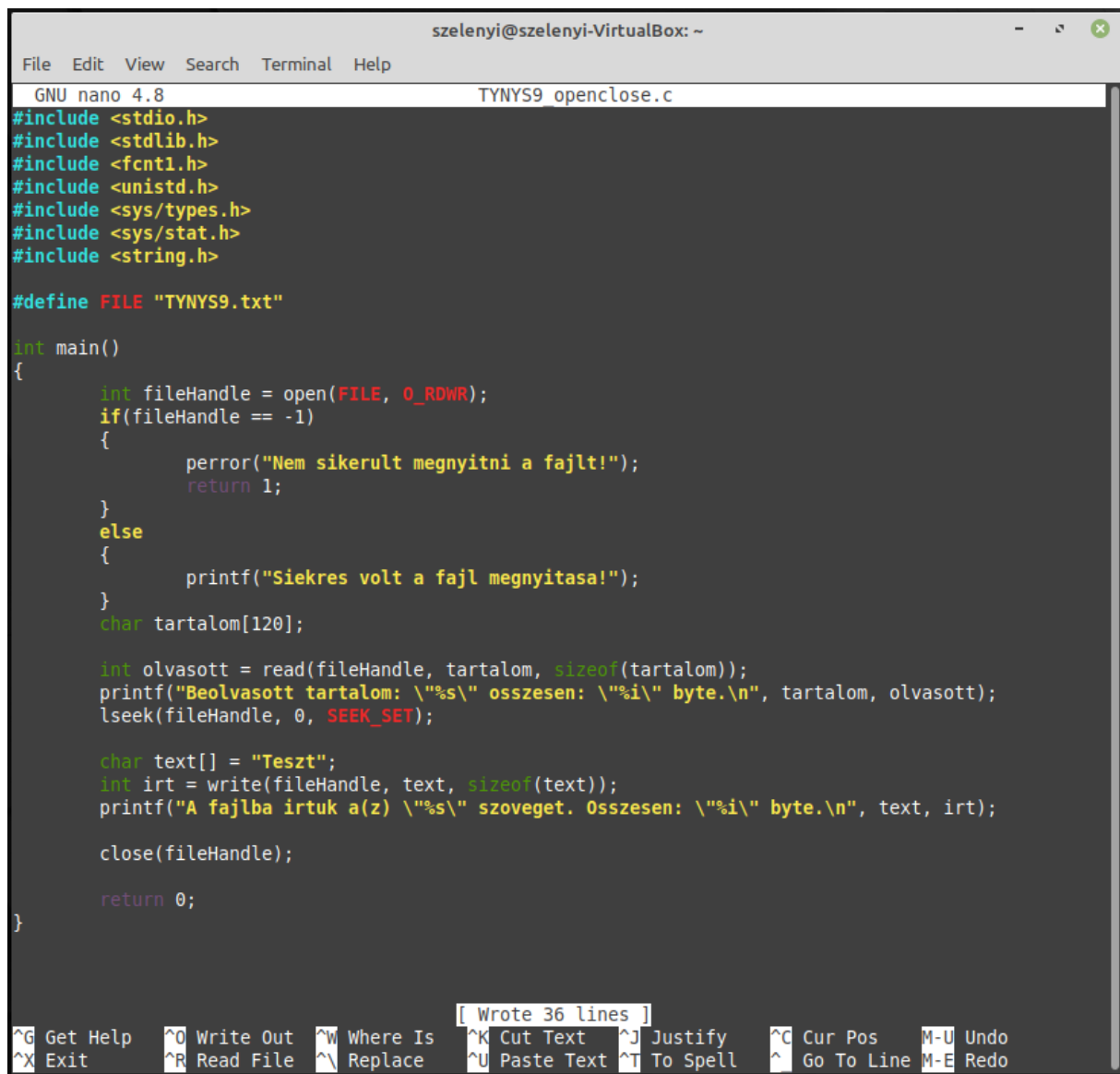
Mérnökinformatikus hallgató

TYNYS9

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.



```
szelenyi@szelenyi-VirtualBox: ~
File Edit View Search Terminal Help
GNU nano 4.8 TYNYS9_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 "TYNYS9.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!");
    }
    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;
}

[ Wrote 36 lines ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line M-E Redo
```

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\_tobbsignal.c

```
szelenyi@szelenyi-VirtualBox: ~
File Edit View Search Terminal Help
GNU nano 4.8 TYNYS9_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;
    }

    for(;;)
    {
        printf("A program leallitasahoz a kovetkezoeket vegezze el: \n");
        printf("1. Nyisson meg egy masik terminalt.\n");
        printf("2. Adj ki a parancsot: kill: %d \n", getpid());
        sleep(10);
    }
    return 0
}

void handleSignals(int signum)
{
    switch(signum)
    {
        case SIGINT:
            printf("\nCTRL + 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 0;
}

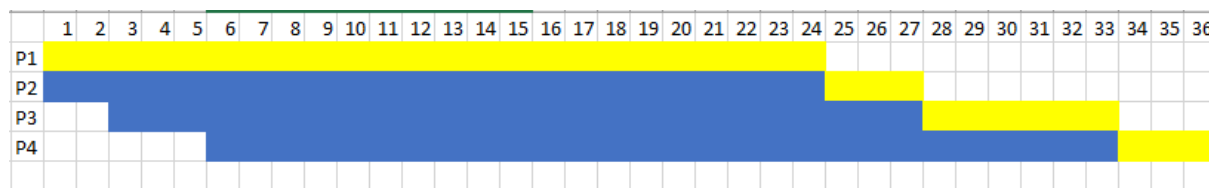
[ Wrote 47 lines ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line M-E Redo
```

	P1	P2	P3	P4
Érkezés	0	0	2	5
CPU idő	24	3	6	3
Indulás				
Befejezés				
Várakozás				

Algoritmus neve	
CPU kihasználtság	
Körülfordulási idők átlaga	
Várakozási idők átlaga	
Válaszidők átlaga	

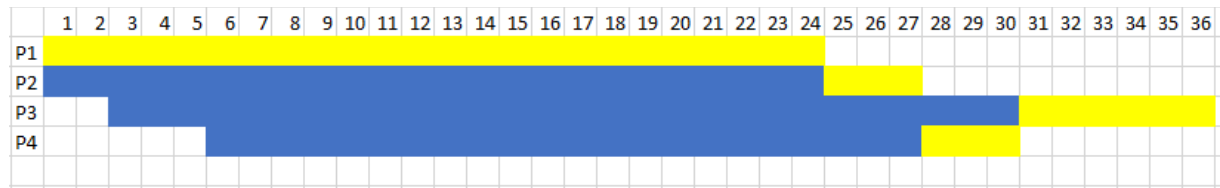
FCFS

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
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			



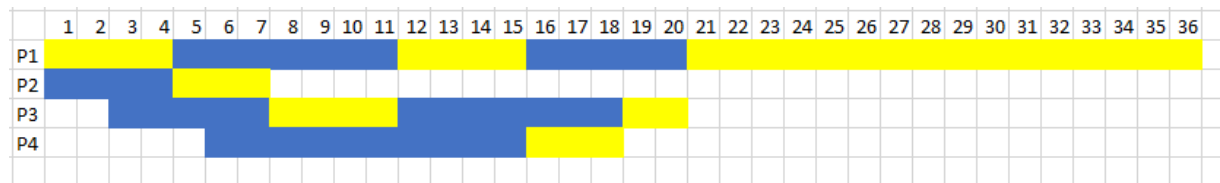
## SJF

SJF	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			

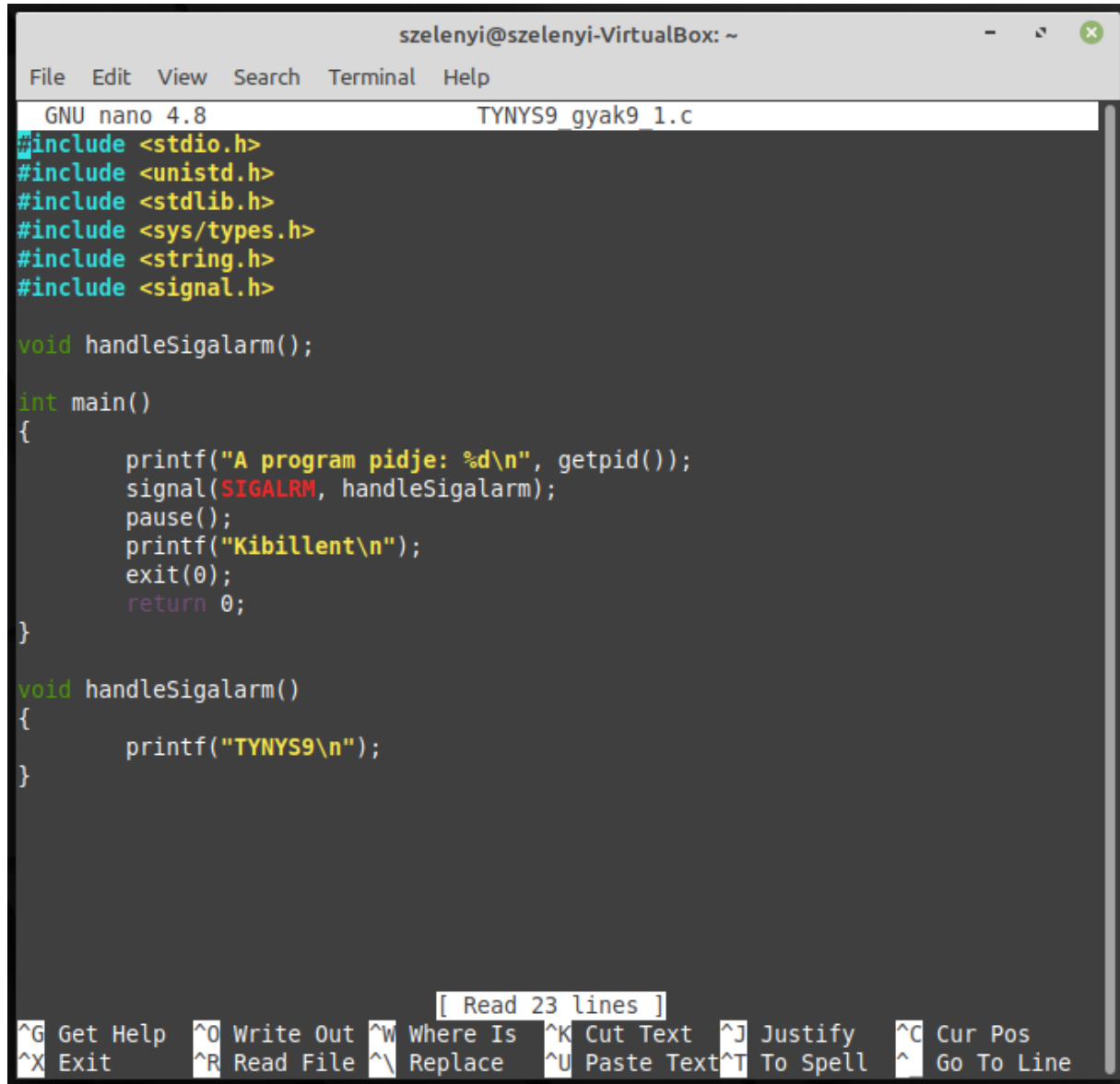


## RR: 4ms

RR: 4ms	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
Körülfordulási idő	4, 11, 21		7 9, 9		13
Algoritmus neve	RR: 4ms				
CPU kihasználtság					156,4
Körülfordulási idők átlaga					10
Várakozási idők átlaga					7
válaszidők átlaga					95,6

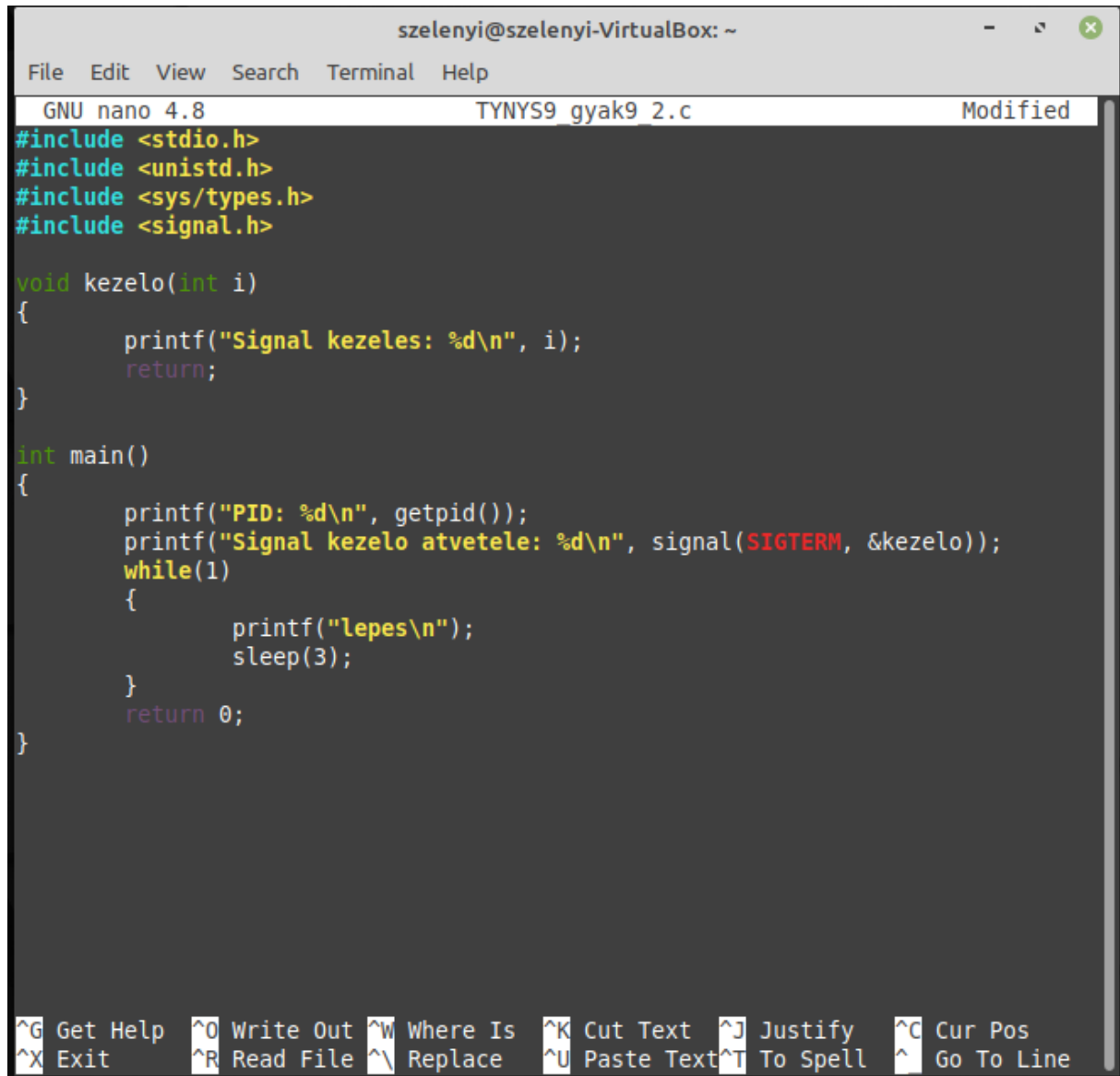


2. Írjon C nyelvű programot, amelyik kill() seg.-vel SIGALRM-et küld egy argumentumként megadott PID-u processznek, egy másik futó program a SIGALRM-hez rendeljen egy fv.-t amely kiírja pl. neptunkodot, továbbá pause() fv.-el blokkolódjon, majd kibillenés után jelezze, hogy kibillent és terminálódjon.



```
szelenyi@szelenyi-VirtualBox: ~  
File Edit View Search Terminal Help  
GNU nano 4.8 TYNYS9_gyak9_1.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("TYNYS9\n");  
}  
  
[ Read 23 lines ]  
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos  
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line
```

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.”



The screenshot shows a terminal window titled 'szelenyi@szelenyi-VirtualBox: ~'. Inside, the GNU nano 4.8 text editor is open, editing a file named 'TYNYS9 gyak9 2.c'. The code in the editor is a C program that handles the SIGTERM signal. It includes headers for stdio, unistd, sys/types, and signal. A function named 'kezezo' takes an integer argument and prints it. The main function prints the PID, registers the 'kezezo' function for SIGTERM, and enters a loop where it prints 'lepes' and sleeps for 3 seconds. The bottom of the window shows a status bar with various keyboard shortcuts like ^G Get Help, ^O Write Out, etc.

```
szelenyi@szelenyi-VirtualBox: ~
File Edit View Search Terminal Help
GNU nano 4.8                                TYNYS9 gyak9 2.c                                Modified
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <signal.h>

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

int main()
{
    printf("PID: %d\n", getpid());
    printf("Signal kezezo atvetele: %d\n", signal(SIGTERM, &kezezo));
    while(1)
    {
        printf("lepes\n");
        sleep(3);
    }
    return 0;
}

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text ^T To Spell  ^_ Go To Line
```