

Operációs rendszerek BSc

5. gyak.

2021. 03. 10.

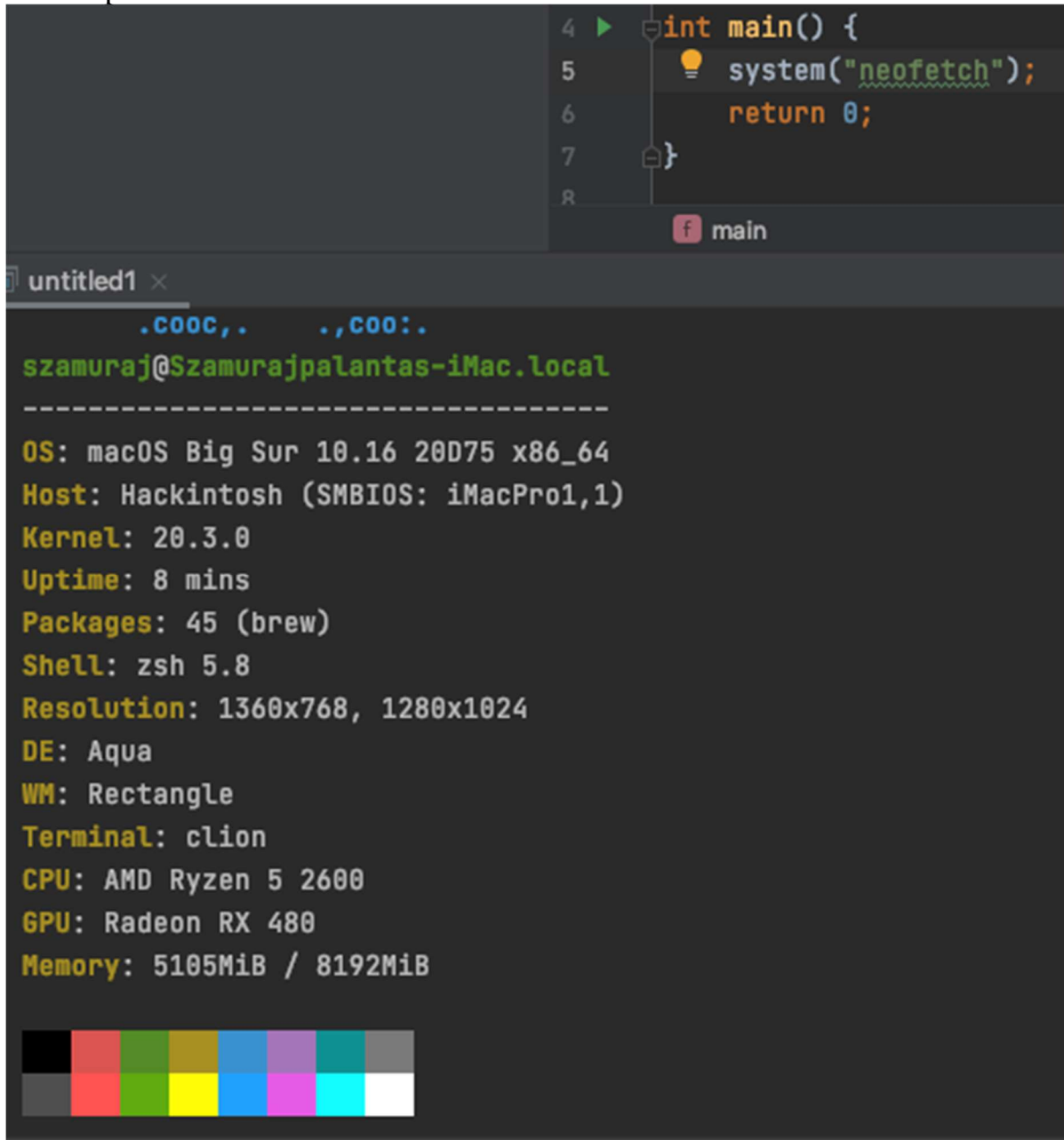
Készítette:

Simonyák János Bsc
Üzemmérnök-informatikus
MZ727W

Miskolc, 2021

1) feladat – system()

a) Létező parancs

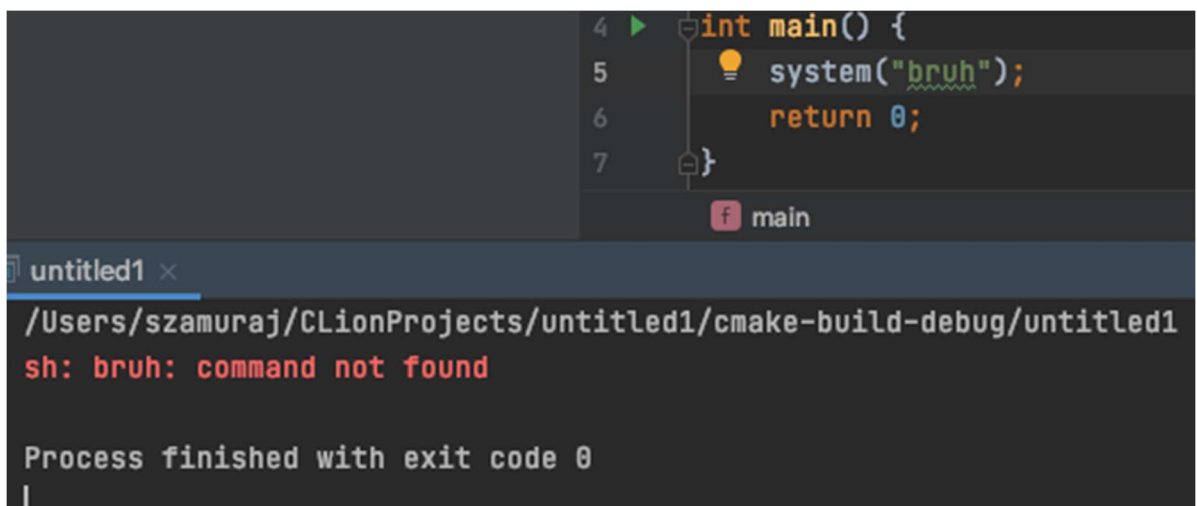


```
4 int main() {  
5     system("neofetch");  
6     return 0;  
7 }  
8  
f main
```

untitled1 x

```
.cooc,. .,coo:..  
szamuraj@Szamurajpalantas-iMac.local  
-----  
OS: macOS Big Sur 10.16 20D75 x86_64  
Host: Hackintosh (SMBIOS: iMacPro1,1)  
Kernel: 20.3.0  
Uptime: 8 mins  
Packages: 45 (brew)  
Shell: zsh 5.8  
Resolution: 1360x768, 1280x1024  
DE: Aqua  
WM: Rectangle  
Terminal: clion  
CPU: AMD Ryzen 5 2600  
GPU: Radeon RX 480  
Memory: 5105MiB / 8192MiB  
  
██████████ ██████████ ██████████ ██████████ ██████████ ██████████ ██████████ ██████████  
██████████ ██████████ ██████████ ██████████ ██████████ ██████████ ██████████ ██████████
```

b) Nem létező parancs

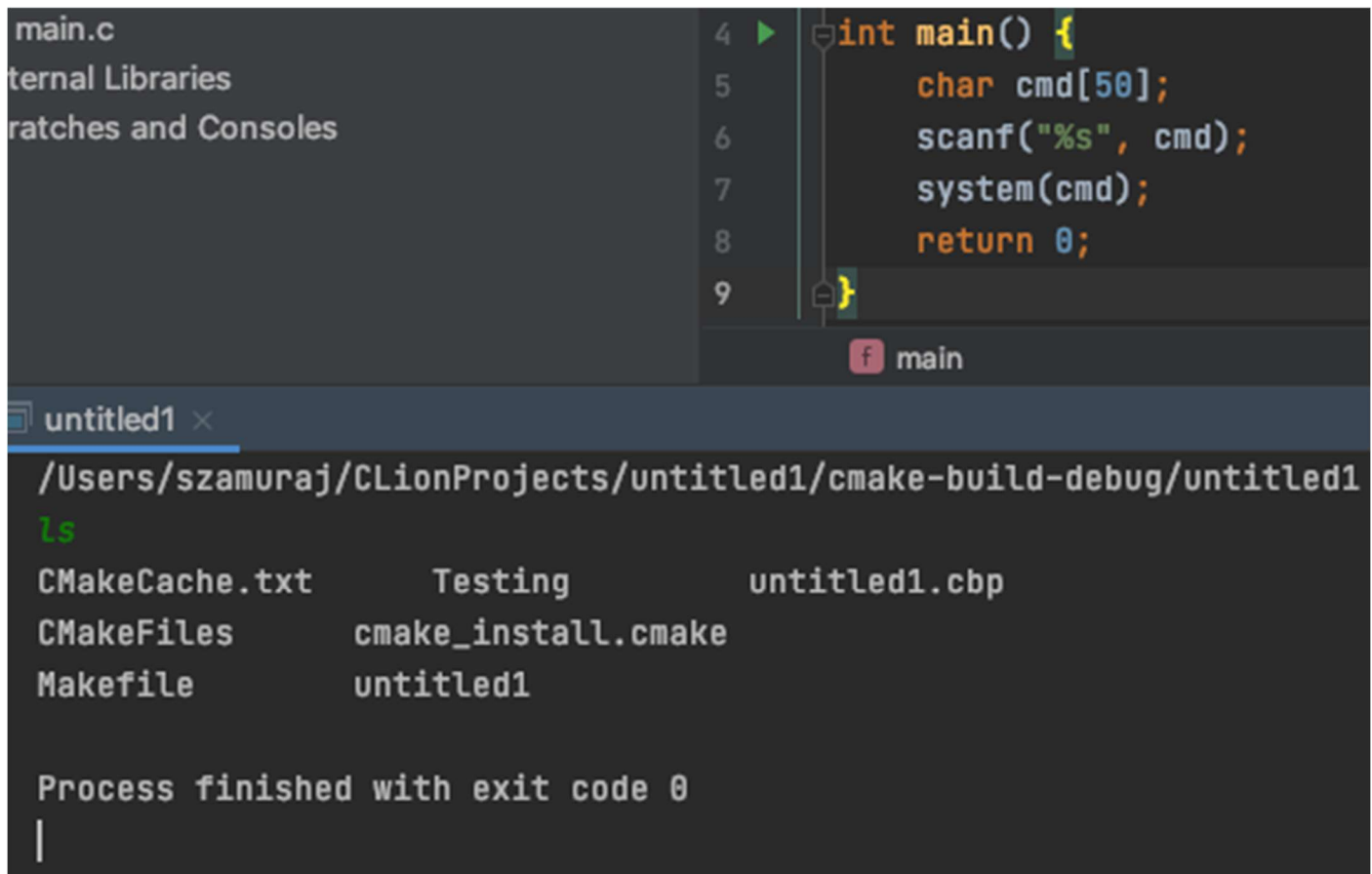


```
4 int main() {  
5     system("bruh");  
6     return 0;  
7 }  
8  
f main
```

untitled1 x

```
/Users/szamuraj/CLionProjects/untitled1/cmake-build-debug/untitled1  
sh: bruh: command not found  
  
Process finished with exit code 0
```

2) feladat – „Terminál emulátor”



The screenshot shows the CLion IDE interface. The top editor displays a C program named `main.c` with the following code:

```
1 // ...  
4 int main() {  
5     char cmd[50];  
6     scanf("%s", cmd);  
7     system(cmd);  
8     return 0;  
9 }
```

The bottom panel shows a terminal window titled `untitled1` with the following output:

```
/Users/szamoraj/CLionProjects/untitled1/cmake-build-debug/untitled1  
ls  
CMakeCache.txt      Testing            untitled1.cbp  
CMakeFiles          cmake_install.cmake  
Makefile            untitled1  
  
Process finished with exit code 0  
|
```

3) feladat – Gyermekfolyamat



The first screenshot shows a C program using `fork()` and `execl()` to create a child process:

```
#include <sys/types.h>  
#include <sys/wait.h>  
#include <stdio.h>  
#include <unistd.h>  
  
int main(void) {  
    int pid;  
    if ((pid = fork()) < 0)  
        perror("fork error");  
    else if (pid == 0) {  
        if (execl("./child", "child", (char *)NULL))  
            perror("execl error");  
    }  
    if (waitpid(pid, NULL, 0) < 0)  
        perror("wait error");  
    return 0;  
}
```

The second screenshot shows a C program using `printf()` and `sleep()` to create a child process:

```
#include <stdio.h>  
#include <unistd.h>  
  
int main(void) {  
    for (int i = 0; i < 5; i++) {  
        printf("Simonyák János, MZ727W\n");  
        sleep(1);  
    }  
    return 0;  
}
```

4) feladat – Gyermekfolyamat – exec

5) feladat – Gyermekfolyamat – Befejezési állapotok

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <unistd.h>

int main(void)
{
    int pid;
    int status;

    if ((pid = fork()) < 0) {
        perror("Fork error");
        exit(7);
    }
    else if (pid == 0)
        abort();

    if(wait(&status)!=pid)
        perror("Wait error");

    if(WIFEXITED(status))
        printf("Successful");

    return 0;
}
```

6) feladat – Ütemezés

RR: 5ms	Érkezés	CPU igény	Indulás	Befejezés	Váró processz	Várakozás	Maradék idő
P1	0	3	0	3	P2	0	-
P2	1	8	3	8	P2, P3	2	3
P3	3	2	8	10	P2, P4	5	-
P2*	(8)	3	10	13	P4, P5	2	-
P4	9	20	13	18	P4, P5	4	15
P5	12	5	18	23	P4	6	-
P4*	(18)	15	23	28	P4	5	10
P4*	(28)	10	28	33	P4	0	5
P4*	(33)	5	33	38	-	0	-

