



Operating systems

Lab. 2.

I. Issues to prepare

- Shells, *bash*
- *bash* scripts
- C programming

II. Outline

1. Getting familiar with *bash* commands
2. Simple *bash* script creating
3. First C project

III. Tasks

1. Start Linux on virtual machine configured during last laboratories. Log in and run terminal.

bash is a UNIX system shell. Useful commands:

pwd – shows present directory

```
student@student1 ~ $ pwd
/home/student
```

ls – lists content of present directory

ls -lh – lists more detailed info

```
student@student1 ~/c $ ls
main.c main.i main.o
student@student1 ~/c $ ls -lh
razem 32K
-rw-r--r-- 1 student student 73 paź 24 12:15 main.c
-rw-r--r-- 1 student student 18K paź 24 12:15 main.i
-rwxr-xr-x 1 student student 4,9K paź 24 12:16 main.o
```



parameter *--help* shows help about present program

man ls – show manual of present program

```
student@student1 ~ $ ls --help
Usage: ls [OPTION]... [FILE]...
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.
```

cd 'folder' – changes present directory to another. You can use relative paths as well as absolute ones.

```
student@student1 ~ $ pwd
/home/student
student@student1 ~ $ cd folder
student@student1 ~/folder $ cd ..
student@student1 ~ $ cd /home/student/folder/
```

mkdir 'folder' – creates directory called *folder*

touch 'file' – creates file called *file*

nano 'file' – edits text file *file*

```
student@student1 ~ $ mkdir folder
student@student1 ~ $ cd folder/
student@student1 ~/folder $ touch pliktekstowy
student@student1 ~/folder $ nano pliktekstowy
```

In text editor nano:

Ctrl + o – saves file

Ctrl + x – exits from file

Back again to *bash*:

cp 'what' 'where' – copies file *what* to directory *where*

mv 'what' 'where' – moves file *what* to directory *where*

rm 'what' – removes file *what*

* - wildcard may be substituted for any set of characters



```
student@student1 ~ $ cp plik1.txt /home/student/folder/
student@student1 ~ $ mv plik2.txt ./folder/
student@student1 ~ $ cd folder/
student@student1 ~/folder $ ls -lh
razem 8,0K
-rw-r--r-- 1 student student 0 paź 24 14:05 plik1.txt
-rw-r--r-- 1 student student 0 paź 24 14:05 plik2.txt
-rw-r--r-- 1 student student 5 paź 24 12:28 pliktekstowy
-rwxr-xr-x 1 student student 55 paź 24 12:51 scr.sh
student@student1 ~/folder $ rm *.txt
student@student1 ~/folder $ ls -lh
razem 8,0K
-rw-r--r-- 1 student student 5 paź 24 12:28 pliktekstowy
-rwxr-xr-x 1 student student 55 paź 24 12:51 scr.sh
```

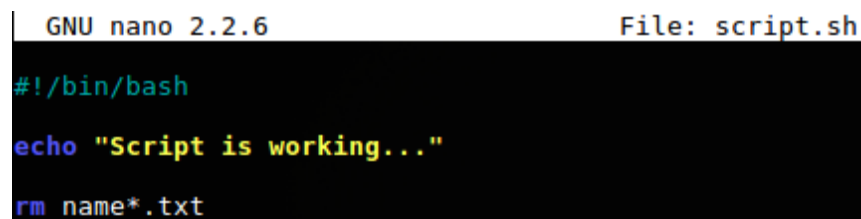
> - redirects standard output to new file

>> - redirects standard output and appends at the end of file

```
student@student1 ~/folder $ ls -lh > lista.txt
student@student1 ~/folder $ cat lista.txt
razem 8,0K
-rw-r--r-- 1 student student 0 paź 24 14:08 lista.txt
-rw-r--r-- 1 student student 5 paź 24 12:28 pliktekstowy
-rwxr-xr-x 1 student student 55 paź 24 12:51 scr.sh
```

2. Inside your home directory (like `"/home/student"`) create the new one in which you will store any source files. There create the new text file with your name as its name. Fill it with some text. Create another file with your name and some number as its name.
3. Prepare *bash* script.

bash scripts are text files with the `.sh` extension in which *bash* commands are listed. Sample script:



```
GNU nano 2.2.6                               File: script.sh
#!/bin/bash
echo "Script is working..."
rm name*.txt
```



Freshly created file does not have any permissions. If you want make the file executable you can use *chmod*:

```
chmod +x script.sh
```

Script may be run using relative path (`./script.sh`) as well as full command like:

```
bash script.sh
```

4. Write script which in present directory creates folder *backup*, then search for all files with your name in name, copy all of them to *backup* dir and finally remove all of them from present dir.
5. Write Hello World! program using *c* language.

C programming algorithm:



Create file *main.c* and write there source code in *c* language. Then run compiler `gcc main.c -o program.o`. Such output file `.o` should be executable like `.sh` scripts.

6. Execute manually all steps of building application, redirect to text files output of pre-processor, assembler and linking units.