

# EXERCISE NO. 1

SUBMISSION DUE DATE: 9/4/2017 23:55

## INTRODUCTION

The purpose of this assignment is to get familiar with Linux and development environments. The assignment includes the following:

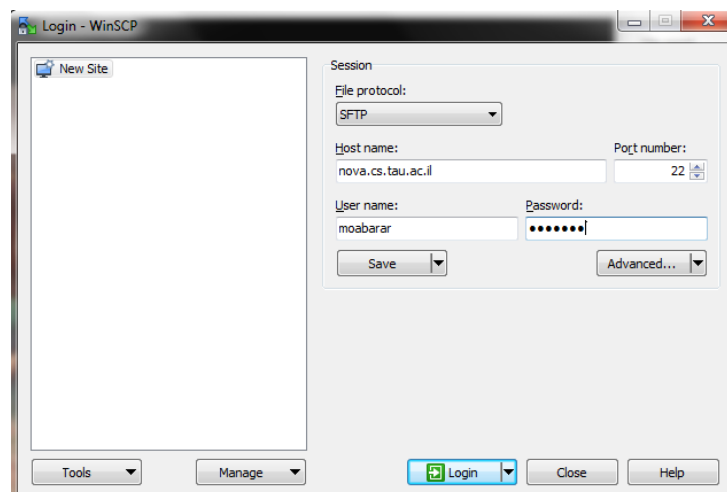
- 1- Remotely connect to the Nova server and get familiar with basic shell commands.
- 2- Create your first C Program.
- 3- Compiling, debugging and basic use of makefiles.

## DEVELOPMENT ENVIRONMENT

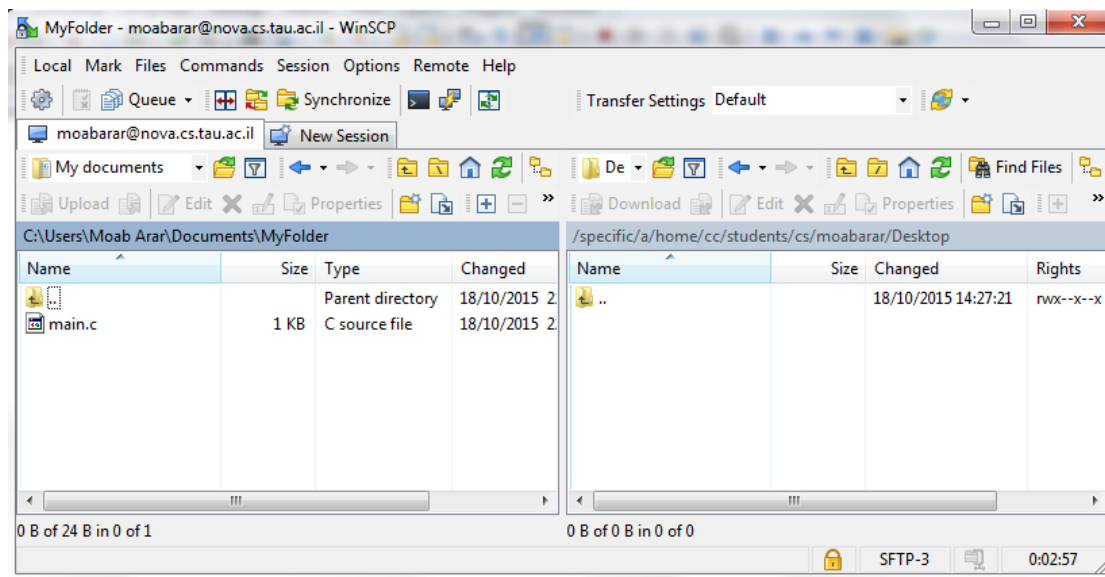
Students may work in any development environment they like. However we encourage you to use eclipse as your development tool (we will not support issues regarding other IDEs or operating systems other than Linux or windows). Your submission will be automatically checked by a script, so your code should run properly on Nova - the faculty server. You can use the computers in the PC farm or remotely connect to the server as will be described below.

## FILE TRANSFERRING

If you need to transfer files from your personal computer to Nova, you can use WinSCP. To connect you will need to enter your personal authentications. See the example below.

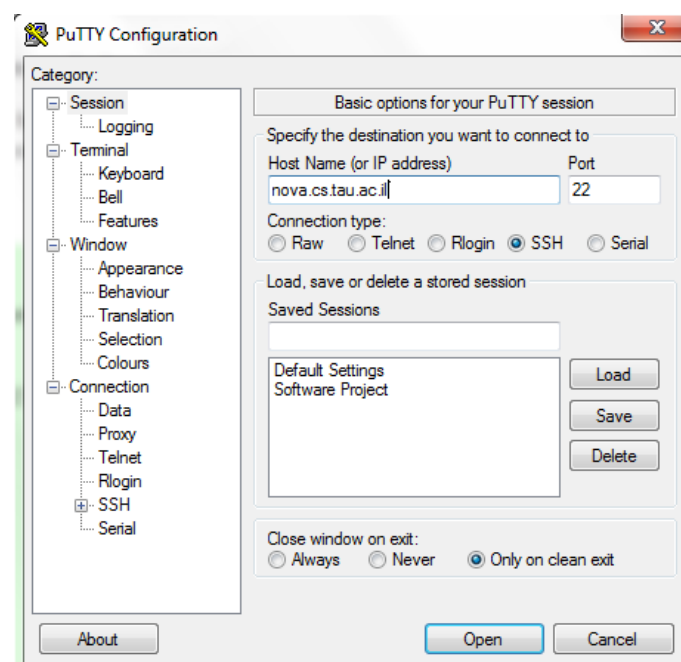


After connecting you can upload your files by dragging the designated file from your personal computer to the directory in the Nova server:

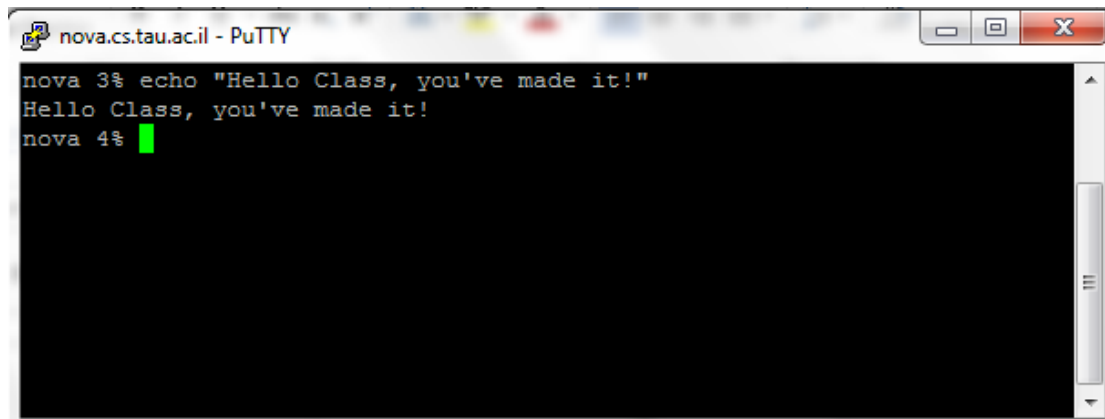


## REMOTE CONNECTING TO NOVA

To connect to the Nova server you can use PuTTY:



After pressing open you will be asked to enter your username and password for authentication. (The username and password are the same as in your moodle authentication.) When successfully connected to Nova you will be able to see the terminal as in the following picture:



```
nova 3% echo "Hello Class, you've made it!"
Hello Class, you've made it!
nova 4%
```

Now you can enter shell commands and execute them by pressing enter. Read through the section below to find out more about shell commands.

(Note: echo is a shell command that is used to print strings to the standard output.)

## USEFUL LINKS

You can download PuTTY and WinSCP by clicking on the [following link](#). Recall that we recommend that you work with eclipse; please follow the installation guidelines for eclipse in moodle.

## BASIC SHELL COMMANDS

After the connection with the server (Nova) is established, you can now type in shell commands and execute them simply by pressing enter. The results will be shown on the terminal (If there's any).

A few useful shell commands:

```
>> pwd
```

Prints the full pathname of the current working directory to the standard output. (The pathname is relative to the root directory which is the first directory in Linux.)

```
>> ls [dir]
```

Lists the content of the directory "*dir*" (Both files and directories). If no parameters are given, the result is the content of the current working directory.

```
>> cd [dir]
```

Changes the current directory to be *dir*.

Use the following shortcuts:

"." – This is a shortcut for the current directory.

".." – This is a shortcut for the parent directory in the hierarchy.

"~" – this is a shortcut for the home directory.

```
>> mkdir [dirName]
```

Creates a new directory with the name *dirName*.

```
>> cp [file1] [file2] ... [fileK] [dir]
```

Copies *file1, file2, ..., fileK* to the directory "*dir*".

Note: In order to copy an entire directory (recursively copy a directory) use `-r` flag.

Examples:

```
>> cp /dir1/myFile /dir2
```

Copies "*myFile*" which is located in "*/dir1*" to the directory "*/dir2*"

```
>> cp -r /sourceDir /destinationDir
```

This copies the directory "*sourceDir*" (with its content) to the directory "*destinationDir*"

```
>> rm [file1] [file2] ... [fileK]
```

Removes *file1, file2, ..., fileK*.

```
>> man [command]
```

The `man` command is used to display the manual page of the command "*command*".

Note: you can navigate through the manual page using the arrows in the keyboard. To exit the manual page press "*q*"

```
>> diff [file1] [file2]
```

Prints the difference between the two files (*file1* and *file2*)

Note: Use the *man* command to see the manual page of the command "*sdiff*".

## FIRST C PROGRAM

In this assignment you will write your first C program. The program receives a number in base *a* and converts it to base *b*. For example if you receive the number 101 in base 2 (binary base) and you are asked to convert it to base 10 (decimal) then the result should be 5.

The user will first enter the base in which the number is represented (base *a*) and then the desired base (base *b*). Afterwards the user inputs the number she wants to convert as a series of chars.

## MAIN

The main function is the entry point of any C program. We will avoid implementing other functions inside the `main.c` file. However, in this assignment, you may implement auxiliary functions inside the source file **main.c**.

The behavior of the main function is as follows:

- 1- First the program receives an integer representing base *a*. The program asks the user to input the base by printing the following message:  
**"Please enter the number's base:\n"**

- a. If the user enters a base  $a \notin [2,16]$  then the program will print the following error and terminates:  
**"Invalid input base\n"**
- 2- Afterwards, the program receives an integer representing base  $b$ . The program asks the user to input the base by printing the following message:  
**"Please enter the desired base:\n"**
  - a. If the user enters a base  $b \notin [2,16]$  then the program will print the following error and terminates:  
**"Invalid desired base\n"**
- 3- The program will ask the user to enter a number in base  $a$  by printing the following message:  
**"Please enter a number in base <a>:\n"**  
 Where <a> is the value of the base  $a$ .  
 For example if  $a = 3$  then the message is:  
**"Please enter a number in base 3:\n"**
  - a. The user will input the number as a series of chars, if at any point the users enters an invalid char (i.e. a character that is not in base  $a$ ) then the program will print the following error and terminates  
**"Invalid number!\n"**
- 4- Finally the program will convert the number from base  $a$  to base  $b$  and print the following message:  
**"The result is : <res>\n"**  
 Where <res> is the number received in (section 3) printed in base  $b$ .

#### Assumptions and requirements:

- You may assume that the input number is a non-negative integer.
- The input number ends when an EOF character is received.
- If at any point there was an error while reading from the standard input then the program prints the following error and terminates:  
**"An error occurred!\n"**

## COMPILE, DEBUG AND MAKEFILE

As stated before, students may work on any development environment they choose. However your code should compile and run on Nova. Please upload your source code (main.c) and the makefile provided in the assignment zip file to nova (note that all the files should be in the same directory).

When the connection is established set your current directory as the source code's (main.c) directory. Type in the following command and press enter in order to build your program:

```
>> gcc -std=c99 -Wall -Wextra -Werror -pedantic-errors main.c -o ex1
```

The following command will compile your program and will create a binary file called ex1. Note that the compilation flags used in the command above will be our default flags, so it is recommended that you configure your eclipse with these flags (more info on moodle).

Now you can run your program by executing the following line on Nova:

```
>> ./ex1
```

See example below:

```
nova 11% ./prog
Please enter the number's base:
16
Please enter the desired base:
10
Please enter a number in base 16:
A4
The result is : 164
```

You can also check your program using the files given in the zip file simply by following these instructions:

- 1- Compile your program.
- 2- Use I/O redirection as follow:  
`>> ./ex1 < test1.in > result1.out`  
this will use "*test1.in*" as the standard input and write the results to the file "*result1.out*"  
(The file will be created if it doesn't exist.)
- 3- Compare your results with the expected result for "*test1.in*", the expected result for "*test1.in*" is "*expected1.out*".  
`>> diff result1.out expected1.out`
- 4- If **nothing** was printed, then your results matches the results we got (Good job). If not, try to debug and find the source of your mistake.
- 5- You can repeat steps 1-4 for test2.in, test3.in etc...

Before submitting your code make sure your code compiles using the makefile provided in the zip file. To do so, follow these instructions:

- 1- Copy your source code along with the file "makefile" into the Nova server. Place the files in the same directory. (If you have already done so in previous round, just ignore this)
- 2- Go the directory where you copied the files to, and simply write the following command:  
`>> make`
- 3- As an output, make sure the files "main.o" and "ex1" were added to the directory (use "ls" command). You can run your program by executing the file "ex1" as stated in previous sections.

**Note:** Please look at the makefile and read the guidelines in the course material for more information on makefile. You will need to implement one on your own in future assignments.

## SUBMISSION

Please submit a zip file named **id1\_id2\_assignment1.zip** where id1 and id2 are the ids of the partners. The zipped file must contain the following files:

- main.c – Your source code for the program.
- partners.txt – This file must contain the full name, id and moodle username for both partners. Please follow the pattern in the assignment files. **(do not change the pattern)**
- makefile – the makefile provided in the assignment files. (Don't change it!)

**Note: Both students should submit the assignment on moodle.**

## REMARKS

- For any question regarding the assignment, please don't hesitate to contact Moab Arar by mail: [moabarar@mail.tau.ac.il](mailto:moabarar@mail.tau.ac.il).
- Late submissions are not acceptable unless you have the lecturer approval.
- Borrowing from others' work is not acceptable and may bear severe consequences.

GOOD LUCK