



SE 2XA3 (2019/20, Term I) Minor Lab 3 -- lab section L03

[Back To Lab Menu](#)
[Back To Main Menu](#)
[Log Out](#)

If you are in the correct lab room and during the SCHEDULED time slot for your lab section for Minor Lab 3 and working on one of the lab's workstations, go into Marks+Comments to see if your attendance was registered. If you signed on the course website too early (the most common problem), your attendance would not be recorded; in such case log out and sign on again. If your attendance is still not recorded, ask your TA to record your attendance and email Prof. Franek right away. If you are not using one of the lab's workstations and want to have your attendance recorded, tell the TA.

The Minor Labs are open book labs, you can bring and use any notes etc., you can access any website, Google, Wikipedia etc. You are allowed to cooperate with other people, inside or outside the lab. The TA can provide extensive individual help as long as this would not deprive other students of the TA's help.

Task 1. Create scripts `s1` and `s2` and `makefile`

Create a bash script `s1` that has 3 command line arguments. If the number of arguments is not correct, it displays a message and terminates. If the number of arguments is correct, it creates a text file with the name given by `$1`, and the content of the file is `$3` lines containing the line number and `$2` separated by a comma. For instance, if `s1 hello bye 100` is executed, a file named `hello` will be created and it will contain 100 lines: `1,bye ... 100,bye`

[a sample solution for script s1](#)

Create a bash script `s2` that has four line arguments. It creates a file whose name is `$2` from a file whose name is `$1`. It creates it by copying from `$1` the lines that contain strings `$3` or `$4`. The last line of the file `$1` should say `done`. For instance, executing `s2 hello test 1 3` will create a file named `test` containing all the lines from the file `hello` that contain `1` or `3` and a line `done`, or `s2 hello test a bc` will create a file named `test` containing all the lines from the file `hello` that contain `a` or `bc` and a line `done`.

[a sample solution for script s2](#)

Prepare the following makefile:

- The file `test1` does not depend on anything. It is made of 300 lines `1,test1 ... 300,test1`. The script `s1` is used to make it.
- The file `test2` depends on `test1`. The file `test2` contains all the lines from `test1` that contain pattern `0,test` or `1,test`. The script `s2` should be used to make it.
- When `make all` is executed, both `test1` and `test2` are created or re-created. So you need to "force" re-creation of `test1` even when it exists and is up-to-date. So make it dependant on some imaginary file for which there is one rule in the makefile -- this imaginary file does not depend on anything and is not build at all (in the sample solution this imaginary file is called `.FORCE`)
- The files `test1` and `test2` remain in the directory after the makefile execution is done.
- When `make clean` is executed, the files `test1` and `test2` are removed.

[a sample solution for makefile](#)

Find out about the LINUX command `script` for recording of a terminal session; the best is to use `man` or google it. Read on how to finish recording of the terminal session, in particular in Bourne (and hence Bash) shell.

Now run the `script` command to create a file named `lab3` that contains the “record” of the following interactive session:

- `make all`
- `echo test1:`
- `cat test1`
- `echo test2:`
- `cat test2`
- `rm test1`
- `sl test1 xxx 50`
- `echo test1:`
- `cat test1`
- `make all`
- `echo test1:`
- `cat test1`
- `echo test2:`
- `cat test2`

After exiting from `script`, try display the content of the file `lab3` by `cat lab3`
Can you explain what was happening at each stage of the session?

[how should lab3 file look like](#)

Task 2. Create `f2.py` and `makefile`

Download this python program `f1.py` and transfer it to *moore*. Just in case, transform it to UNIX text file by running `dos2unix f1.py`. Try to execute the python program by executing `python f1.py`; you will get an error for the `f2.py` module is missing.

Your task is to write a makefile that does two things, (i) creates the module `f2.py`, and then (ii) executes `f1.py`. Note, that when you execute a python program `f1.py`, a file `f2.pyc` is automatically generated.

The makefile should be executed in the silent mode, i.e. `make -s f1`

The output produced should look like

`This is a nice Python program`

`Sat`

`This is the end of this nice Python program`

where the second line should be the current date (Mon, Tue, Wed, Thu, Fri, Sat, or Sun).

The module `f2.py` created by the makefile should contain

```
# module f2.py
def day():
    x=" Sat Sep 28 11:37:18 EDT 2019 "
    x=x[1:4]
    print x
```

where the value of the string stored in `x` is produced by the `date` command. Note that the next line, `x=x[1:4]` will extract the name of the current day, in this example it would be `Sat`

So, `f1` must depend on `f1.py` and `f2.py` and is 'made' by executing `f1.py` by `python f1.py`

`f2.py` must depend on an imaginary file (so it is forced to be created) and is 'made' by several echo commands that will create the file.

When `make clean` is executed, `f2.py` and `f2.pyc` are removed.

[a sample solution for makefile](#)