B: SE 2XA3 (2018/19, Term I) Major Lab 1 -- lab section L01

Back To Lab Menu | Back To Main Menu | Submissions | Log Out

sample solutions: `script1` and `script2`

If you are in the correct lab room and during the SCHEDULED time slot for your lab section for Major Lab 1 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, please log out and sign on again. If your attendance is still not recorded, please, ask your TA to record your attendance and email Prof. Franek right away. If you are not using one of the lab's workstations, please, have the TA record your attendance.

The Major Labs are open book labs, you can bring and use any notes etc. You can access any website, Google, Wikipedia etc. The only thing that is not allowed is cooperation with other people, inside or outside the lab. You must submit your own work. The TA can only help with administrative and technical aspects and not with the solutions of the problems of the Major Lab.

In this lab, there are three tasks and three deliverables: text files `safety` and `script1` and `script2`. Each can be submitted either via the course website, or using `2xa3submit`. For `2xa3submit` submission please use `2xa3submit AAA proj1 BBB` where `AAA` is your student number and `BBB` the name of the file you want to submit. You can submit every file as many times as you wish, the latest submission will be used for marking. The submission is opened from 8:30-11:20 or 14:30-17:20 depending on the lab section; after the closing time no submission is be possible. If submission is not possible for whatever reason (typically right after the submission closes), email the file or files as an attachment immediately (needed for the time verification) to Prof. Franek (franek@mcmaster.ca) with an explanation of the problem; you must use your official McMaster email account for that. Include the course code, your lab section, full name, and your student number. Note that if there is no problem with submission (typically the student using a wrong name for the file), you might be assessed a penalty for email submission, depending on the reason you used email.

Task 1. Safety Quiz

Please, type your answers to a text file named **safety** . The safety manual can be found in the Help section.

The safety quiz questions:

1. Who is in your case the Departmental Laboratory Supervisor?
2. Name three important rights of Laboratory Workers.
3. Name three important responsibilities of Laboratory Workers.
4. Can you bring food to the lab?
5. Can you bring drinks to the lab?
6. Can you plug/unplug the lab workstations from the electric outlet?
7. Can you bring and use your laptop in the labs?
8. Can you use an extension cord to connect your laptop in the labs?

Task 2. bash script named `script1`

The name of your bash script must be **script1** and below is a description of what it should do when executed.

1. All displayed messages must be exactly as shown here, with all details. All commands must follow the order described below.
2. The script creates 4 text files named **file_one**, **file_two**, **file_three**, and **file_four** .
3. The content of **file_one** are three lines `First line of file_one` and `Second line of file_one` and `Third line of file_one`, the content of **file_two** are three lines `First line of file_two` and `Second`

`line of file_two` and `Third line of file_two`, ... , the content of **file_four** are three lines `First line of file_four` and `Second line of file_four` and `Third line of file_four` .

4. Then the script displays the line `content of file_one` followed by the contents of the file, followed by the line of equal signs.
5. Similarly for **file_two** ... **file_four** .
6. Then contents of all the files **file_one** .. **file_four** are concatenated into a single file named **JOINTFILE** in the following order: first **file_four**, then **file_one**, then **file_two** and then **file_three** .
7. The script displays the line `content of JOINTFILE` and then displays the contents of the file **JOINTFILE** .
8. Then the script displays line `listing of current directory` and lists the contents of the current directory.
9. Then it removes all the files **file_one**, **file_two**, **file_three**, **file_four**, and **JOINTFILE** .
10. Then the script displays the line `listing of current directory` and lists the contents of the current directory.
11. At this point the script terminates.
12. After the execution of the script, the current directory is in the same state as it was when the script started its execution.

A sample run:

```
content of file_one
First line of file_one
Second line of file_one
Third line of file_one
=======================
content of file_two
First line of file_two
Second line of file_two
Third line of file_two
=======================
content of file_three
First line of file_three
Second line of file_three
Third line of file_three
=======================
content of file_four
First line of file_four
Second line of file_four
Third line of file_four
=======================
content of JOINTFILE
First line of file_four
Second line of file_four
Third line of file_four
First line of file_one
Second line of file_one
Third line of file_one
First line of file_two
Second line of file_two
Third line of file_two
First line of file_three
Second line of file_three
Third line of file_three
listing of current directory
file_four  file_three  JOINTFILE  script2
file_one   file_two    script1
listing of current directory
script1  script2
```

3. bash script named `script2`

The name of your bash script must be **script2** and below is a description of what it should do when executed.

1. All displayed messages must be exactly as shown here, with all details. All commands must follow the order described below.
2. The scripts displays `creating file Garbage` and creates a file named **Garbage** in the current directory. The file contains a single line
   `*** garbage ***` .
3. The scripts displays `creating Aux_Dir` and creates a directory named **Aux_Dir** in the current directory.

4. Then the script displays a line `moving Garbage to Aux_Dir` and moves the file **Garbage** to the directory **Aux_Dir** .
5. Then the script displays `listing of Aux_Dir` and then list the contents of **Aux_Dir**.
6. Then the script displays `displaying Garbage` and then list the contents of the file **Garbage** in **Aux_Dir** .
7. Then the script stores in a variable a string obtained by piping the output of **ls -la** to **grep** applied to **Aux_Dir** . The first 10 letters of the string contain the permissions for the directory **Aux_Dir** . Play with **ls -la Aux_Dir** to see what it produces. To store the result of **ls -la | grep Aux_Dir** in a variable **var** use **var=`ls -la | grep Aux\_Dir`** (note the back quotes !). To extract the first 10 characters from the string in **var** for the use in **echo** in the next step, use **${var:0:10}** .
8. Then the script displays a single line `original permissions of Aux_Dir: xxx` where **xxx** is the permissions (the permissions must be obtained from the directory, not hard coded in the script !!!).
9. Then the script displays line `change it to d---rwxrwx` and using **chmod** it changes the permissions to the desired combination. Then as in 7. and 8. , it displays a line containing `new permissions of Aux_Dir: xxx` where **xxx** is the changed permissions (the permissions must be obtained from the directory, not hard coded in the script !!!).
10. Then the script displays line `change it to drwxr--r--` and using **chmod** it changes the permissions to the desired combination. Then as in 7. and 8. , it displays a line containing `new permissions of Aux_Dir: xxx` where **xxx** is the changed permissions (the permissions must be obtained from the directory, not hard coded in the script !!!).
11. Then the script displays line `change it to drwx--x--x` and using **chmod** it changes the permissions to the desired combination. Then as in 7. and 8. , it displays a line containing `new permissions of Aux_Dir: xxx` where **xxx** is the changed permissions (the permissions must be obtained from the directory, not hard coded in the script !!!).
12. Then the script displays line `trying to remove Aux_Dir by rm` and using **rm** it tries to remove the directory (it should not work).
13. Then the script displays line `trying to remove Aux_Dir by rmdir` and using **rmdir** it tries to remove the directory (it should not work).
14. Then the script displays line `trying to remove Aux_Dir by rm -r` and using **rm -r** it tries to remove the directory (it should work) at which point the script terminates.
15. After execution of the script, the current directory should not contain the directory `Aux_Dir` nor the file `Garbage`.

A sample run:

```
creating file Garbage
creating Aux_Dir
moving Garbage to Aux_Dir
listing of Aux_Dir
Garbage
displaying Garbage
*** garbage ***
original permissions of Aux_Dir: drwxr-xr-x
change it to d---rwxrwx
new permissions of Aux_Dir: d---rwxrwx
change it to drwxr--r--
new permissions of Aux_Dir: drwxr--r--
change it to drwx--x--x
new permissions of Aux_Dir: drwx--x--x
trying to remove Aux_Dir by rm
rm: cannot remove `Aux_Dir': Is a directory
trying to remove Aux_Dir by rmdir
rmdir: Aux_Dir: Directory not empty
trying to remove Aux_Dir by rm -r
```