COMP 206 - Winter 2018 - Assignment 0

Goal to provide an introduction to the programming environment we will use during the term, ensure you have the correct accounts and access needed to use the McGill computing resources we will use for the rest of the assignments.

Not for marks, but required as you need access to GitHub to get the course material and to know how to submit to My Courses to complete the assignments.

Question 1 - Our Course GitHub

Create your account on https://github.mcgill.ca. This is done by simply navigating to this address using your browser, and typing in your McGill Campus Wide Login (the one used for Minerva and My Courses). As long as you're registered for the course or the waitlist, you should have the ability to create an account.

Find the repository for this assignment: https://github.mcgill.ca/COMP206Winer2018/Assignment0. Take a look at the files through the Github web interface. You will be adding your own file to this repository.

Using a Linux shell to operate git from the command-line, add a file of your own to the repository by following the "clone, edit, add, commit, push" cycle described during Lecture 2. From your Linux terminal, these will be the steps:

1. Add your ssh keys to GitHub by:
    1. Running "ssh-keygen" in your Linux shell. It is fairly safe to leave your passphrase blank here and just keep pressing enter until the process ends
    2. Selecting settings from the menu at the top-right of the Github webpage
    3. Entering the SSH and GPG keys section by selecting that option from the left panel
    4. Clicking "New SSH Key" on the top right of that page
    5. Enter your ssh public key into the box, by opening the file "~.*ssh*id_rsa.pub" in Linux and copying the contents into the Github webpage
    6. Give your key a name you can remember, such as "SOCS"
    7. Click add SSH key
2. Clone the repository, by typing "git clone git@github.mcgill.ca:COMP206Winter2018/Assignment0.git" into the Linux terminal
3. You should now have an Assignment0 folder in that directory, enter it by typing "cd Assignment0" into the terminal

4. View the files in that folder with “ls”, “cat”, or by using your favorite text editor
5. Add a new file to the directory, using a text editor.
    1. The response file you add should follow the specification that you can find in my response, david_meger.txt. That specification defines that the file name is your first and last name separated by an underscore, with the “.txt” extension. Copy the first line of david_meger.txt into your file, so it’s identical. Write you own content for the 2nd line, making sure to separate each answer using commas in between them. You should not use any commas within the answer for each question, since the commas are used to separate answers.
    2. Note that you are not required to provide your real name, since this repository is public and anyone with access to this GitHub server will be able to see our responses. Later when we submit code for real assignments with grades attached, we will use private repositories that are secure. Also, the answers for each question are just for fun, so feel free to make something up, or leave a blank simply by having only spaces between two commas.
6. Inform git to be aware of your changes by typing “git add <your file name>”
7. Create a new revision with your changes by typing “git commit -m “<your log message>”
8. Push the changes back to the github server by typing “git push”
9. There are 2 possible outcomes.
    1. The push completes without error, you are done and can now see your file through the Github page
    2. The push outputs text like “error: failed to push some refs... hint: Updates were rejected because the remote contains work that you do not have locally”. This means another student pushed in between your clone and push. You need to get a fresh copy and push again, with these commands:
        1. “git pull”
        2. “git merge”
        3. “git add <your file name>”
        4. “git commit -m “<your message>”
        5. “git push”

Question 2: Hello, world

Ensure that you can create a file named “hello.c”, compile it using the command “gcc hello_world.c” and run with the command “./a.out” to see “Hello, world.” printed on the

terminal as was shown during Lecture 2.

Once your code works by running manually, use our automated tester, which is found inside the same Assignment 0 repository you cloned in Question 1, within the “eval” folder. Run the tester by typing “python eval.py <path to folder that contains your code>”. You will see output on the terminal indicating the outcome. We will often provide you testers because they help you ensure you match the **coding specification** perfectly. Once your code passes all 3 tests, it is quite likely close to a good solution. Do note though, we will run additional private tests during marking, so it is not sufficient for you to simply read the tester and write code that only passes those exact cases. You must always think about writing code that is robust and will satisfy the specification for all cases requested.

Submit your hello.c file on My Courses, under the Assignments tab and in the Assignment 0 Handin folder. This assignment is not marks, but we will run our automated testing on all code submitted, and in case we find any problems with the way you submitted, we’ll let you know to prevent problems later.