Assignment #0

Due Date: Friday, September 17, 2021, 5:00 pm EST

This assignment is designed to get you familiar with the most basic aspects of working with Linux, and with assignment submission. **It is not worth any marks, but you must get 100% on this assignment to get credit for the other assignments.**

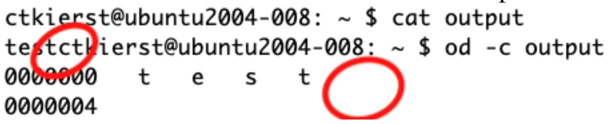
Topics that must have been completed before starting:

1. Linux: The Teaching Environment
2. Linux: Interacting with the Shell
3. Linux: Directories and Files
4. Handouts: Getting Started
5. Handouts: Linux Commands

If you've never logged into the CS undergraduate student environment before, you will need to set your password first (this is *not* the same as your Quest password). Follow step 1 of the "Getting Started Guide".

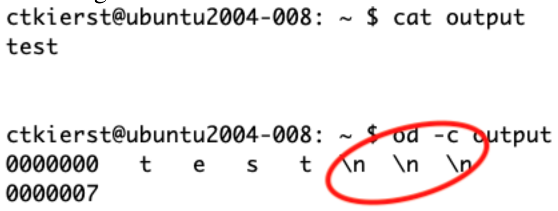
For any question that asks you to make sure your file ends with a newline character (think A1 questions 19, 20, etc.) you should probably not be pressing Enter at the end of the line as long as you're editing the file on a Linux machine (e.g. the student environment). Most Linux text editors automatically insert a newline character at the end of the file by convention. `wc` will not count these end-of-file newline characters as their own lines.

A good method of testing whether your file has the correct number of new lines is by running the command `cat` on the file, and observing the location of the next command prompt. Say we create a file `output` that just contains the text "test". If the next command prompt appears on the same line as the output of the command `cat output`, i.e, "testuserid@ubuntu1804-004: $", then there are no new lines at the end of your file, and the file format is incorrect. For example, you can see that there is no `\n` shown after the word "test" in the output of the `od -c` command:



```
ctkierst@ubuntu2004-008: ~ $ cat output
testctkierst@ubuntu2004-008: ~ $ od -c output
0000000   t   e   s   t
0000004
```

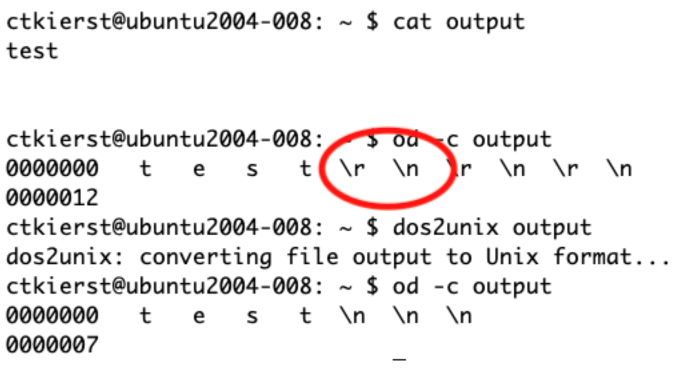
If there are empty lines after "test" and before the next command line prompt, then there are too many new lines at the end of your file, and this file format is also incorrect. For example, the output of `od -c` shows that there are 3 instances of `\n` following the word "test":



```
ctkierst@ubuntu2004-008: ~ $ cat output
test


ctkierst@ubuntu2004-008: ~ $ od -c output
0000000   t   e   s   t  \n  \n  \n
0000007
```

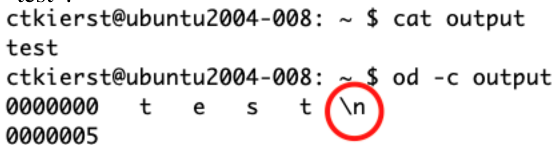
Your text editor may also use MS DOS line endings, which will appear as `\r\n` when displayed using the `od -c` command, as in:



```
ctkierst@ubuntu2004-008: ~ $ cat output
test


ctkierst@ubuntu2004-008: ~ $ od -c output
0000000   t   e   s   t  \r  \n  \r  \n  \r  \n
0000012
ctkierst@ubuntu2004-008: ~ $ dos2unix output
dos2unix: converting file output to Unix format...
ctkierst@ubuntu2004-008: ~ $ od -c output
0000000   t   e   s   t  \n  \n  \n
0000007
                              _
```

You want just one new line at the end of your file, which will give you "test" on one line, and "userid@ubuntu1804-004: $" on the very next line. For example, you can see from the following that `od -c` shows only a single `\t` following the word "test":



```
ctkierst@ubuntu2004-008: ~ $ cat output
test
ctkierst@ubuntu2004-008: ~ $ od -c output
0000000   t   e   s   t  \n
0000005
```

1. Read the course outline at `https://www.student.cs.uwaterloo.ca/~cs246/F21/outline.shtml`
2. Log into your `linux.student.cs.uwaterloo.ca` account and execute the command `ls`. You should see a directory named `cs246`. If you do not see this directory, create it via the command: `mkdir cs246`
3. Navigate to your `cs246` directory: `cd cs246`
4. Verify that you are in your `cs246` directory: `pwd`
5. Clone the course GIT repository, which only needs to be done once:

   ```
   git clone ssh://linux.student.cs.uwaterloo.ca/u/cs246/pubrepo/1219/.git
   ```

   (You will use the command `git pull` to obtain updates of the repository contents. To use it successfully, you must be in your `cs246/1219` directory *before* issuing the `git pull` command.)
6. Verify that the checkout succeeded: `ls`. You should see a directory called `1219`.
   (Parenthetical note: 1219 is Quest-speak for Fall 2021. The last digit is the month, and the first three digits, added to 1900, give the year.)
7. Verify that you are still in the `cs246` directory and NOT in the `1219` directory: `pwd` (your should be in `~/cs246`)
8. Throughout the term, the instructors will add new files to the repository, which will be added to your `cs246/1219` directory. However, you should create your assignment solutions in a different directory, so the changes you make to the files do not conflict with the changes made by the instructors. So, you should create an empty directory to work on. We suggest naming it `f21`, but feel free to use whatever name you want: `mkdir f21`.
9. Navigate to the newly created directory: `cd f21`.
10. Copy the A0 files from the GIT directory to your working directory: `cp -r ../1219/a0 ./`
11. Navigate to the assignment 0 directory: `cd a0`
12. Once again, verify that you are in the correct directory: `pwd` (you should be in the directory `~/cs246/f21/a0`)
13. Using a text editor (either vi or emacs), create the file `hello.txt`, with contents exactly as shown below:

```
Hello from Linux!
I used vi.
```

If you used emacs, replace `vi` above with `emacs`. **You should press enter at the end of the first line, and at the end of the second line.** Once you have created the file, use the `wc` command to determine how many lines the file contains. Take note of the relationship between the number of times you pressed Enter, and the number of lines contained in the file. The exact result will depend on your editor. (You may find using `od -c` on your file instructive.)

14. Navigate to your home directory: `cd` (or `cd ~`)

15. List the hidden files in your home directory: `ls -d .*`

16. Determine whether your home directory contains a file called `.bash_profile` — if it doesn't, `cp .profile .bash_profile`; if it does, move on to the next step. If you're missing the `.profile` file, you can copy the one from `/etc/skel/.profile` over first, and then proceed.

17. Using a text editor (either vi or emacs), open the file `~/.bash_profile` (`vi ~/.bash_profile` or `emacs ~/.bash_profile`). This file should not be empty; if it is, check that you have typed the name of the file correctly. Add the following lines to the *end* of this file:

    ```
    source ~cs246/setup
    source ~cs246/setup2
    alias g++14="g++ -std=c++14 -Wall -g"
    ```

    (Optional) We recommend also adding the following lines to the end of this file:

    ```
    alias vi="vi -X"
    export EDITOR=vi
    ```

    If you choose to use vi, these lines will make vi launch faster, and will ensure that other tools (like git) default to vi when they launch a text editor. If you choose to use emacs, omit the first line, and replace `vi` with `emacs` in the second line. Save your changes and exit (in vi, hit Escape and type `:wq`, followed by Enter; in emacs, Ctrl-X, Ctrl-S, Ctrl-X, Ctrl-C).

18. Navigate to your `a0` directory: `cd cs246/f21/a0`.

19. Using a text editor (either vi or emacs), create the text file `path1.txt` that contains the answer to the following question: if your current directory is `/u/jdoe/cs246/1219`, what *relative path* is equivalent to the *absolute path* `/u/jdoe/cs246/1219/lectures/c++/overload`? (Note that the question mark at the end of the previous sentence is *not* part of the path, it's just the punctuation for the question.) Make sure, as always, that your file ends with a newline character (whether this implies that you must press Enter will depend on your editor). Use `wc` to verify for yourself that your file consists of exactly one line.

20. Using a text editor (either vi or emacs), create the text file `path2.txt` that contains the answer to the following question: if your current directory is `/u/jdoe/cs246/1219`, what *relative path* is equivalent to the *absolute path* `/u/jdoe/cs245/a1`? (Note that the directory name consists of the letter 'a' followed by the digit '1' (one), not a lower-case letter 'L'.) Make sure, as always, that your file ends with a newline character. Use `wc` to verify for yourself that your file consists of exactly one line.

21. Read the manual page for the `wc` command: `man wc`.

22. Use `wc` to count the number of *words* (and only the number of words, not the number of lines or number of characters) in your file `hello.txt`, and use output redirection to store the result in the file `hellowords.txt`.
    (Note that you must use output redirection to solve this question; you will likely fail the test if you manually create a file with the number of words.)

23. Create a text file called `promise.txt` that contains the following text, all on one line:

```
I promise not to publicly ask for or provide hints about Marmoset test cases
or assignment solutions on Piazza.
```

24. Make a zip file containing all of the files in your a0 directory: `zip a0.zip *` — **make sure you are in your** `cs246/f21/a0` **directory when you do this**, otherwise your file will contain your entire `a0` directory structure, and not just the files contained in a0. (Having the directory structure will cause you to fail the Marmoset tests.)

    If you're re-submitting to Marmoset, you'll need to re-zip your submission, i.e. run the command to create a new ZIP file again (**this applies to all assignments!**) A ZIP file just stores files inside of it–it won't automatically reflect changes you make to files outside of it! (Note that if your ZIP file contains files that it shouldn't, you'll need to delete the ZIP file using the `rm` command before zip'ing again; otherwise, it just adds the (new) files to the existing ZIP file, replacing any files with the same name.)

25. Read this document about submitting assignments to Marmoset:

    - `http://www.student.cs.uwaterloo.ca/~cs246/current/marm_sub/index.html`

26. Submit the file `a0.zip` to Marmoset `a0`.