CS Homework 0

J. Holly DeBlois

January 26, 2026

1. Use the CS Server

We cover basic concepts/practices/commands that are used at a Linux command-line prompt and in C programming. There are lots of online/print resources available. Here is what CS IT provides:

- https://www.cs.umb.edu/~ghoffman/linux/linux_help.html
- https://www.cs.umb.edu/~ghoffman/linux/nano_text_editor.html
- https://www.cs.umb.edu/~ghoffman/linux/common_unix_commands.html
- https://www.cs.umb.edu/~ghoffman/linux/unix_essentials.html
- https://www.cs.umb.edu/~ghoffman/linux/remote_access_windows.html
- https://www.cs.umb.edu/~ghoffman/linux/remote_access_mac.html
- https://www.cs.umb.edu/~ghoffman/linux/unix_cs_students.html

Note that the fourth item named unix_essentials is indeed essential! Note that type of laptop you have in class matters - do you have windows? mac? linux? - because commands will differ.

2. Login to the CS Server

- From your laptop, use your CS username and password to access the CS servers. This hw0 assumes you have already accessed the CS Department portal and created your course directory. Accessing the portal can be done from your browser by typing: `https://www.cs.umb.edu` and clicking on Portal Login in upper right.
- In this document, we use `bobz` as anyone's cs username. To login, replace bobz with your cs username and type: ssh bobz@users.cs.umb.edu. You will be prompted to enter your password.
- If you have trouble with login, use your UMB email to write to: operator@cs.umb.edu to request help. Please copy your instructor (jane.deblois@umb.edu) on any email you write to operator.
- After you are "on" the CS server that listens for your ssh request, type: exit to leave the server.

3. Simple commands: `pwd`, etc.

Many commands print out basic information. In the example, ¿¿¿ represents a prompt. Run:

- `whoami` to print your username
- `pwd` to print the working directory
- `hostname` to print the name of the host you're currently on
- `w|more` to print out who is on the server (output not shown)

```
>>> whoami
bobz
>>> pwd
/home/bobz
>>> hostname
someserver
```

- a user's prompt often contains helpful information and can be customized by editing, e.g., the `~/.bashrc` file in the user's home directory
- info on command `foo` can be viewed on its "man" page by typing `man foo`; try typing man w
- get a little "meta" by typing `man man`

4. Output file for Homework 0 is output.txt

Entering the commands in this "assignment" will result in a single output file being built line-by-line. At the end, we will compare it to the instructor-provided example output with a `diff` command. In the example, bobz will be replaced by your cs username. Let's start. Run:

- `cd` with no arguments to change to your home directory
- `cd csNNN` where NNN is 444 or 410 to change to your course directory
- `mkdir hw0` to create a directory for the this assignment
- `cd hw0` to change into it
- `touch output.txt` to create the output file we will build
- `ls -la` to view things, which should look something like:

```
>>> ls -la
drwx------ 2 bobz bobz 4096 Sep  2 19:50 .
drwx------ 3 bobz bobz 4096 Sep  2 19:50 ..
-rw------- 1 bobz bobz    0 Sep  2 19:50 output.txt
```

- note that the file is initially empty, i.e., 0 bytes (`man touch` for more)
- note that the ls command has many switches and can do a lot; see `ls`, e.g., here:

  https://www.gnu.org/software/coreutils/manual/html_node/What-information-is-listed.html

5. Building, Step 1: output redirection

We will use output redirection to form the first few lines of our file. Review file handles, redirection, pipes, and related concepts, e.g., here: https://en.wikipedia.org/wiki/Redirection_(computing). Now:

- print a message to stdout with `echo "Hello World!"`
- redirect the message to our file with `echo "Hello World!" > output.txt`
- do both (using tee -a command to write with append) with `echo "Hello again!" | tee -a output.txt`
- append to our file with `echo "my username is:" >> output.txt`
- and again with `whoami >> output.txt` ... the screen output should be looking like this:

```
>>> echo "Hello World!"
Hello World!
>>> echo "Hello World!" > output.txt
>>> echo "Hello again!" | tee -a output.txt
Hello again!
>>> echo "my username is:" >> output.txt
>>> whoami >> output.txt
```

- check contents with `cat output.txt` and number of lines with `wc -l output.txt`
- you should have 4 lines in your file
- to start over at any point, use the one-liner `rm -f output.txt && touch output.txt`
- to be ready for Step 3 below, exit and login again in 2 terminal windows – do your work in one and type: top in the other (the top cmd will monitor all system processes)

6. Building, Step 2: text editor

We can run a text editor from the command line interface (CLI) to add to our file. We will illustrate with `nano`. In class, we will practice using `emacs`, since it is famous in server work. Now:

- examine your PATH variable by typing: `echo $PATH`
- determine the location/existence of the executable of the editor by typing: `which nano`
- is the `nano` executable's location contained in the PATH?
- type executable name's first two characters: `na` and tap `TAB` twice — what happens?
- type `nan` and hit `TAB` — what happens?
- type `nano ou` and hit `TAB` — what happens?
- run `nano output.txt` to add a single (unique, non-empty) line of your choosing at the end of the file; save (CTRL/O and ENTER), exit (CTRL/X)
- edit again, this time running `nano` by its full path `/usr/bin/nano output.txt`; append another (unique, non-empty) line of your choosing, save, exit
- the version of a program is often easy to get at the CLI — type `nano -V`
- get help with `nano -h` or `nano --help` or `man nano`
- check the number of lines in our file with `wc -l output.txt` — you should have 6
- fyi, our file's final version will have 11 lines
- review newline subtleties, e.g., here: https://en.wikipedia.org/wiki/Newline

7. Building, Step 3: (dis)assembly, another login

Run:

- `head -n 3 output.txt > part_A.txt` to copy the first three lines
- `tail -n 3 output.txt > part_C.txt` to copy the last three lines
- `tail -n 4 output.txt | head -n 1 > part_B.txt` to copy just the third line
- `rm output.txt` to delete the original output file
- `cat part_*.txt > new.txt` to assemble the parts
- `uniq new.txt > output.txt` to recreate the output file with non-duplicate lines
- `wc -l output.txt` to check number of lines – should have 6
- `rm -f part_*.txt && rm -f new.txt` to clean up

Now — assuming you got top running in another session earlier, you can use ps to snapshot current processes. In your main session, run the following replacing `bobz` with your username:

- `ps -u bobz` (sample screen output is shown below)
- `ps -u bobz | grep top >> output.txt` to capture your `top` command's process id, etc. in your output file
- `cat output.txt` to see contents and `wc -l output.txt` to see number of lines, now 7

```
  PID TTY          TIME CMD
...
11130 ?        00:00:00 sshd
11132 pts/0    00:00:00 bash
13554 ?        00:00:00 sshd
13555 pts/1    00:00:00 bash
17854 pts/0    00:00:00 top
17855 pts/1    00:00:00 ps
```

8. Building, Step 4: compiling a substitute `wc`

The `wc` ("word count") system command counts the words (or lines, or characters) in the file it is passed as an argument (see `man wc`). We will compile a very small C program that amounts to a *crude* version of this program. Note that **our version will read from standard input**. Now run:

- `q` and then `exit` to close the `top` command/window from before (if still open)
- `wget https://www.cs.umb.edu/~hdeblois/hw0/crude_wc.c` to download the source code file to current directory (`hw0`)
- `less crude_wc.c` to view its contents and `q` to exit the `less` command
- `gcc -o crude_wc crude_wc.c` to compile
- `ls -la` to view directory contents — should look something like this:

```
total 36
drwx------ 2 bobz bobz  4096 Sep  3 02:43 .
drwx------ 3 bobz bobz  4096 Sep  3 02:46 ..
-rwx------ 1 bobz bobz 16664 Sep  3 02:43 crude_wc
-rw------- 1 bobz bobz   635 Sep  3 02:43 crude_wc.c
-rw------- 1 bobz bobz    57 Sep  3 01:48 output.txt
```

- run `wc output.txt` to perform counts with the system `wc`
- run `./crude_wc < output.txt` to perform counts with our crude substitute and compare
- run `./crude_wc < output.txt >> output.txt` three times in a row
- `cat output.txt` and make sense of what you see
- run `ldd crude_wc` to view the shared library dependencies of the executable

9. Building, Step 5: symbolic links

A symbolic link can be useful. Command `ln` makes a hard link. Command `ln -s` makes a soft link. Links are somewhat similar to pointers – they can get broken. Now run:

- `mkdir somedir` to create a subdirectory and change into it with `cd somedir`
- `ln -s ../crude_wc crude_wc_ptr` to create a symlink to our new executable
- `ln -s ../output.txt` to create a symlink to our output file (note lack of 2nd argument)
- `ls -la` should show something like:

```
total 8
drwx------ 2 bobz bobz 4096 Sep  3 03:50 .
drwx------ 3 bobz bobz 4096 Sep  3 03:38 ..
lrwxrwxrwx 1 bobz bobz   11 Sep  3 03:39 crude_wc_ptr -> ../crude_wc
lrwxrwxrwx 1 bobz bobz   13 Sep  3 03:50 output.txt -> ../output.txt
```

- run `echo "this is a test" | ./crude_wc_ptr` to test running the link-to-executable
- run `./crude_wc_ptr < output.txt` to run link-to-executable on link-to-output-file
- run `chmod 770 crude_wc_ptr` and use `ls -la` on both current and parent directories to discern which permissions have changed (note that this behavior may be OS-dependent)
- run `readlink -f crude_wc_ptr >> output.txt` to dereference the link-to-executable and append its full path to our output file
- run cat output.txt to see path added

10. Last tasks

Finally:

- remove the two symlinks with `rm crude_wc_ptr` and `rm output.txt`
- change to parent directory with `cd ..`
- remove empty directory with `rmdir somedir`
- verify that output file has 11 lines with `wc -l output.txt`
- download instructor's output file with
  `wget https://www.cs.umb.edu/~hdeblois/hw0/sample_output.txt`
- run `diff -y output.txt sample_output.txt` to perform a side-by-side comparison of your output file with the instructor's; make sense of any differences