**CSc 3320: Systems Programming**

Spring 2021

Homework

# 2: Total points 100

Submission instructions:

1. Create a Google doc for each homework assignment submission.
2. Start your responses from page 2 of the document and copy these instructions on page 1.
3. Fill in your name, campus ID and panther # in the fields provided. If this information is missing in your document TWO POINTS WILL BE DEDUCTED per submission.
4. Keep this page 1 intact on all your submissions. If this *submissions instructions* page is missing in your submission TWO POINTS WILL BE DEDUCTED per submission.
5. Each homework will typically have 2-3 PARTS, where each PART focuses on specific topic(s).
6. Start your responses to each PART on a new page.
7. If you are being asked to write code copy the code into a separate txt file and submit that as well.
8. If you are being asked to test code or run specific commands or scripts, provide the evidence of your outputs through a screenshot and copy the same into the document.
9. Upon completion, download a .PDF version of the document and submit the same.

Full Name: Noah Platt

Campus ID: Nplatt1

Panther #: 002310477

**Part 1:**

1. Grep uses basic regular expressions, egrep uses extended regular expressions, and fgrep uses strings instead of regular expressions. For example, if the phrase ‘and|or’ were used in grep, the program would search the file for the string ‘and|or’, and ignore the special character. Egrep would read that regex and search the file for occurrences of ‘and’ and occurrences of ‘or’. Fgrep would read ‘and|or’ as a single string as well, as it doesn’t use regular expressions. It wouldn’t understand any special characters such as ‘.’ or ‘\*’. While grep would read ‘\(abc\)\*’ as a string of abc repeated any number of times, fgrep would read it as the string ‘\(abc\)\*’.
2. The tar command compresses and decompresses files. Multiple files can be compressed into a single file by compressing the directory the files are in. For example, if I wanted to compress all of the files in the /Homework2 directory, I would call tar -cvf Homework2.tar Homework2. All files below in hierarchy to Homework2 will be compressed into Homework2.tar.
3. Both the awk and sort commands can break a line into fields using a separator. The default separator is a space ‘ ‘, although that can be changed using the -t option when calling the command. For example, calling awk -t: {print $1} (filename) will call awk and use the colon as a separator to print the first field. Calling sort -t, (filename) will call sort and use the comma as a separator to determine what field should the sort be based on.
4. The sort command sorts the lines of a file. It does this by separating the field based on a separator, chooses a field to sort the lines, then sorts the file based on the field. Any separator can be chosen, including commas, colons, semi-colons, spaces, letters, or numbers. The command also has the ability to specify which field is chosen. For example, the command sort -t, -k2 (filename) will separate each line of the file into fields separated by commas, then sort the lines using the second field.

**Part 2a**

1. Hello World!!!
2. The first command prints the 5th item of each line in the file, as it prints the 5th item of lines which 1 is less than or equal to the number of fields. The second command would print the 1st field of all lines whose line number is greater than or equal to 5. The third command would print all of the 1st through 5th lines. The last command would simply print the 1st field of all lines.
3. good
4. awk ‘/\+$/ {print $0}’
5. The first command would be sed -i 1,5d foo. The second command would be sed -n -e :a -e ‘1,5!{P;N;D;};N;ba’.

**Part 2b**

1. The command prints the line number and first word of every sentence that has a word ending in ‘ing’. The output would be

1:Wish

3:When

4:Now

1. The command looks through the float file and prints the line with the line number greater than 2 and less than 4, with the line number in front of it. The output would be the third line of float, which would look like this.

3:When everything seemed clear.

1. The command reads the file and prints the first field and last field of each line in the file. The output would print “Start to can file” first, followed by the first word and last word in each line, followed by “END- float”.

Start to scan file

Wish,is

strong,,days

When,clear.

Now,all...

END- float

1. The command replaces every space with a tab. The output would be the float file with every space in the file replaced by a tab, which would look like this.

Wish I was floating in blue across the sky, my imagination is

strong, And I often visit the days

When everything seemed so clear.

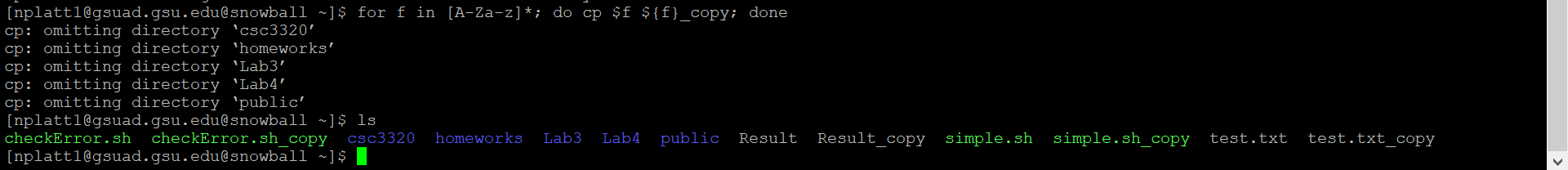
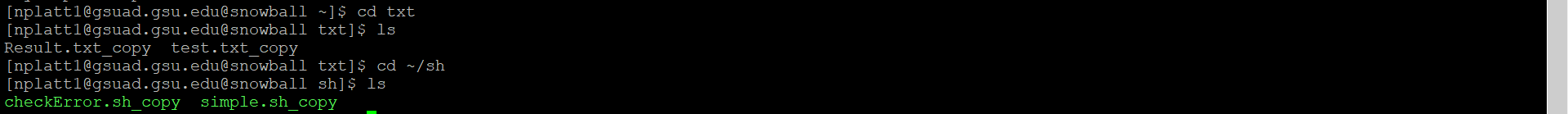
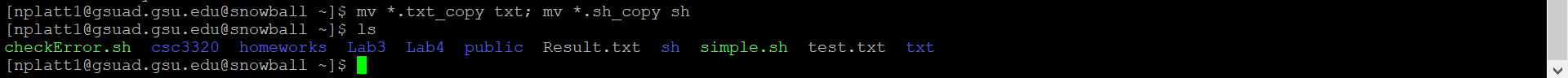
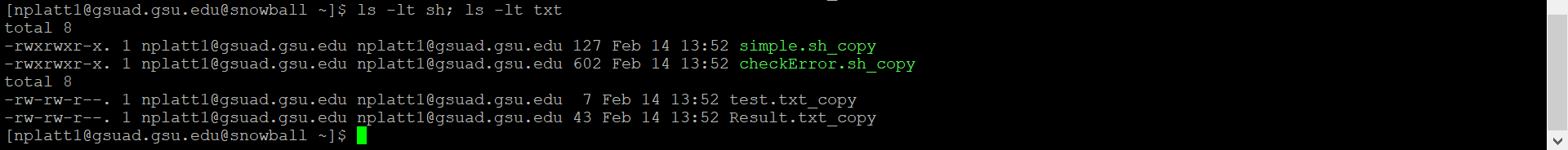
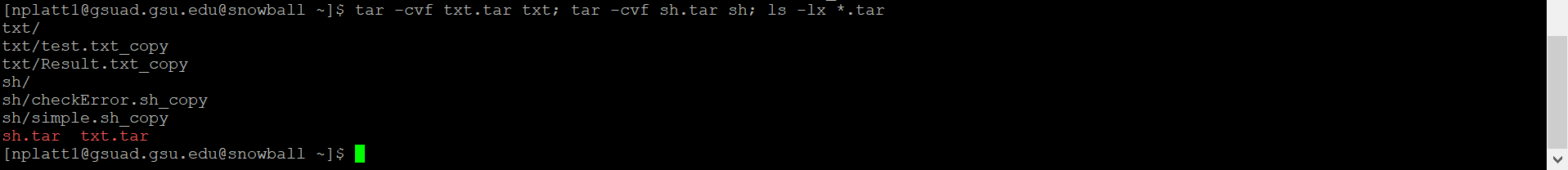
Now I wonder what I’m doing here at all…

1. This searches the awk files for the word BEGIN and prints the line with BEGIN highlighted. The output is

BEGIN {print “Start to scan file”}

1. The code is supposed to create a test directory, two directories within the test directory, a file within the test directory, then enter the test directory and create copies of the subdirectories. The code does not print anything, but outputs the directories test1.bak and test2.bak.

**Part 3**

1. Sorting files
   1. Code used to create copies was “for f in [A-Za-z]\*; do cp $f ${f}\_copy; done
   2. Created new directories with “mkdir txt” and “mkdir sh”, then use “mv \*.txt\_copy txt; mv \*.sh\_copy sh” to move files into the directories.
   3. Used the command “ls -lt txt; ls -lt sh” to sort the files in the directory by month.
   4. Used the command “tar -cvf txt.tar txt; tar -cvf sh.tar sh; ls -lx \*.tar” to create the archive files and list them in ascending alphabetical order. Using -r in the ls command would list them in reverse alphabetical order.
   5. Used the command “tar -cvf copy.tar sh.tar txt.tar” to create the archive of the tar files.