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: Venkata Mani Mohana Rishitha Srikakulapu

Campus ID: vsrikakulapu1

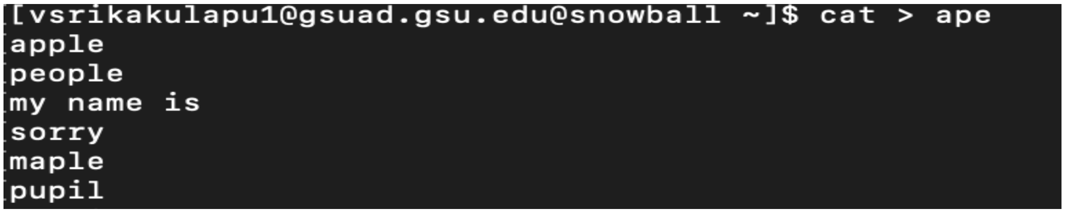
Panther #: 002523638

PART 1 (2.5 points each): 10pts

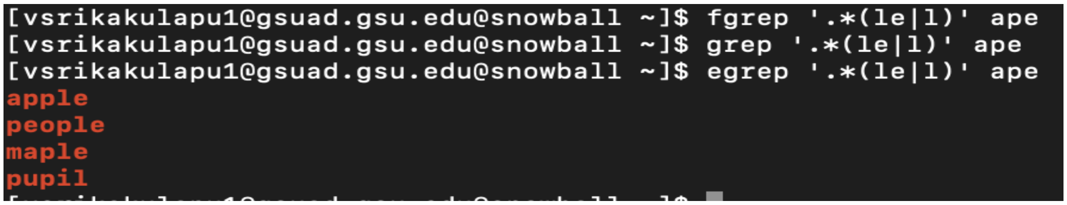
1. What are the differences among ***grep***, ***egrep*** *and* ***fgrep***? Describe using an example.

Grep,egrep and fgrep are used for filtering files. grep searches for basic regular expressions, egrep searches extended regular expressions, fgrep searches only fixed strings.

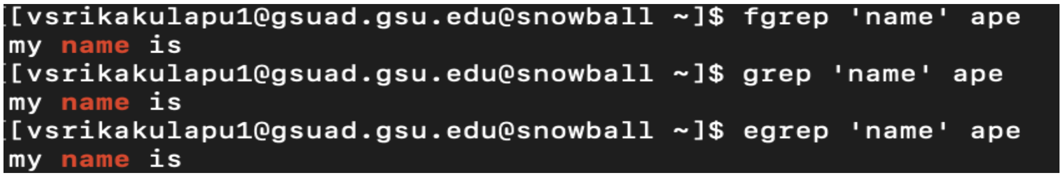
Eg: When we have the below input in a file named ape, we get output for fgrep only for fixed strings and for extended regex patterns, we definitely need egrep and for basic regular expressions grep is useful.



```
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ cat > ape
apple
people
my name is
sorry
maple
pupil
```



```
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ fgrep '.*(le|l)' ape
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ grep '.*(le|l)' ape
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ egrep '.*(le|l)' ape
apple
people
maple
pupil
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$
```



```
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ fgrep 'name' ape
my name is
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ grep 'name' ape
my name is
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ egrep 'name' ape
my name is
```

2. Which utility can be used to compress and decompress files? And how to compress multiple files into a single file? Please provide one example for it.

Archiving (tar) utility can be used to compress and decompress files. Multiple files can be compressed into a single file using the tar command as follows:

$ tar -cvf tarFileName fileList

Eg: $ tar -cvf labWork.tar Lab

Where option -c is to create a tape archive, -f allows us to give a file name to the new tar file, -v stands for verbose.

3. Which utility (or utilities) can break a line into multiple fields by defining a separator? What is the default separator? How to define a separator manually

in the command line? Please provide one example for defining the separator for each utility.

Awk and sort utilities can break a line into multiple fields by defining a separator. The default separator for awk is tabs/spaces. Blank space is the default field separator for sort. Separators can be defined manually in the command line by -

```
$ awk  -F:  '{print $0}'  file.txt
```

-F option is used to manually define a separator for awk utility. Here : (colon) is used.

```
$ sort -t:  +0 -3 file.txt
```

-t option is used to manually define a separator for sort utility. Here : (colon) is used.

4. What does the ***sort*** command do? What are the different possible fields? Explain using an example.

The sort command is used for sorting purpose. It sorts the contents of the file based on ASCII values. However, the default sort command can be given extra options to make it more useful.
Syntax : sort -tc -r {sortField -bfMn}* {fileName}*
where
-t option specifies the field separator here c is the field separator.
-r options reverses the order of default sorting order, i.e., descending order is returned as output instead of ascending order if -r is used.
sortField option specifies the range of sorting field with the help of index + start to - end.
-b option is used to eliminate the leading blanks in the given input
-f option ignores case of the input while sorting
-M option is used to sort according to 3 letter month name
-n option performs numerical sort of the sort field

{fileName}* - input all the files to be sorted

Eg: the sort command is as follows : sort +6 -7 newvalues.txt

This sort command sorts the 6th field of file newvalues.txt where index begins from 0.

Part IIa (5 points each): 25pts

5. What is the output of the following sequence of bash commands: **echo 'Hello World' | sed 's/$/!!!/g'**

   Output is : **Hello World!!!**

   As the echo outputs **Hello World** which is taken as input by the sed command (pipe mechanism) , sed substitutes the end of the input with **!!!**

6. What is the output for each of these awk script commands?

   -- 1 <= NF { print $5 }

   For every line that contains more than or equal to one field, print out the 5th field of that line.

   -- NR >= 1 && NR >= 5 { print $1 }

   For line numbers 5 and greater than 5, the command prints out the first field i.e. in each particular line the first word till it reaches an empty space is printed.

   -- 1,5 { print $0 }

   Prints out every line in the given file.

   -- {print $1 }

   Prints the first field from each line of the given input file.

7. What is the output of following command line:

   echo good | sed   '/Good/d'

   Output is : **good**

   As the echo outputs **good**, it is given as input to sed. Sed command deletes **Good** from the input ,however, the input contains only **good** which is different from **Good**, output remains same i.e. **good**.

8. Which **awk** script outputs all the lines where a plus sign + appears at the end of line?

/\+$/{print $0} - is the **awk** script that outputs all the lines where a plus sign + appears at the end of line.

9. What is the command to delete only the first 5 lines in a file "foo"? Which command deletes only the last 5 lines?

```
$ sed ‘1,5 d’ foo
```

The above command deletes only the first 5 lines in the file "foo"

```
$ sed “$(($(wc -l < foo)-5+1)), $ d” foo
```

The above command deletes only the last 5 lines in the file "foo" where wc -l < foo gives the number of lines in file foo.

Part IIb (10pts each): 50pts

Describe the function (5pts) and output (5pts) of the following commands.

9.

```
$ cat  float
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...
$ cat   h1.awk
NR>2 && NR<4{print  NR  ":"   $0}

$ awk   '/.*ing/ {print NR  ":"  $1}'  float
```

Output is -

```
:Wish
:When
:Now
```

First we use output redirection (>) to the **float** file using cat command. We get the above output because the awk command searches for the pattern which contains any character (except newline character) zero or more times followed by **ing**. For each current line that contains the matched pattern, awk prints **:** followed by the first field of that current line of the file **float,** in which we gave the above input using cat command.

10. As the next command following question 9,

```
$ awk  -f  h1.awk  float
```

Output is -

```
3:When everything seemed so clear
```

We already wrote the program in h1.awk using cat command and redirecting output. Now, in this command, the program to be performed on file **float** are specified in **h1.awk** file and are given to awk using -f option. The program in h1.awk is that for current line number greater than 2 and current line number less than 4, print the current line number followed by **:** followed by the entire line. The current line number greater than 2 and less than 4 is 3 so we get the output as -

3:When everything seemed so clear

11. $cat h2.awk

```
BEGIN { print “Start to scan file”}
{print $1   “,” $NF}
END {print “END-” , FILENAME }
```

$ awk -f h2.awk float

Output is :

```
Start to scan file
Wish,strong,
And,days
When,clear.
Now,all...
END- float
```

In this awk command, the program file is h2.awk. This program has a begin which prints **Start to scan file** in the first line, then in the following lines, the **first field** followed by **,** followed by the **last field** is printed. Then, in the next line **END-** followed by the filename **float** is printed.

12. sed 's/\s/\t/g' float

The output is -

```
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...
```

The sed command substitutes all the whitespace characters and replaces them by tab in the float file.

13. $ ls *.awk| awk '{print "grep --color 'BEGIN' " $1 }' |sh *(Notes: **sh file** runs file as a shell script . $1 should be the output of* ‘ ls *.awk ‘ in this case, not the 1st field *)*

The output is -

```
    BEGIN { print "Start to scan file"}
```

First, ls .*awk lists all the .awk files present in the home directory, we have h1.awk and h2.awk in the home directory. This output is then given to the awk command using pipe (|). This awk command prints out the following :

```
grep --color BEGIN h1.awk
grep --color BEGIN h2.awk
```

i.e, it basically prints the text given in quotes to it followed by each first field. This output is then executed using sh which executes the input as a shell script i.e, it prints out the pattern matching the line with **BEGIN** in color from the 2 files. However, h1.awk doesn't contain BEGIN in it but h2.awk contains and that line is outputted with **BEGIN** highlighted in different color.

14.

```
$ mkdir  test   test/test1  test/test2
$cat>test/testt.txt
This is a test file ^D
   $ cd  test
   $ ls  -l . | grep '^d' | awk '{print "cp  -r  "  $NF  "  " $NF ".bak"}' | sh
```

Here we start by creating a directory named test and 2 directories named test1 and test2 inside the test directory. In the test directory, we also create a test.txt file in which we input the text - 'This is a test file' and exit from the file. We then move into the test directory by using cd command.

In the next command, we long list the contents in the test directory, which contains 2 directories test1, test2 and a testt.txt file. Then we input it to grep to find directories by using the pattern to extract the lines beginning with a 'd'. The output is then given to awk command which prints cp followed by -r followed by last field of output of grep command followed by last field of output of grep command with a .bak tag in addition to it and so the output is as following :

```
cp -r test1 test1.bak
cp -r test2 test2.bak
```

This output is then given as input to sh which runs this as a shell script and does the task of recursively copying the directory test1 to test1.bak and test2 to test2.bak. And so when we list out the contents in the test directory we can see 4 directories that are test1, test1.bak, test2, test2.bak and a file testt.txt.

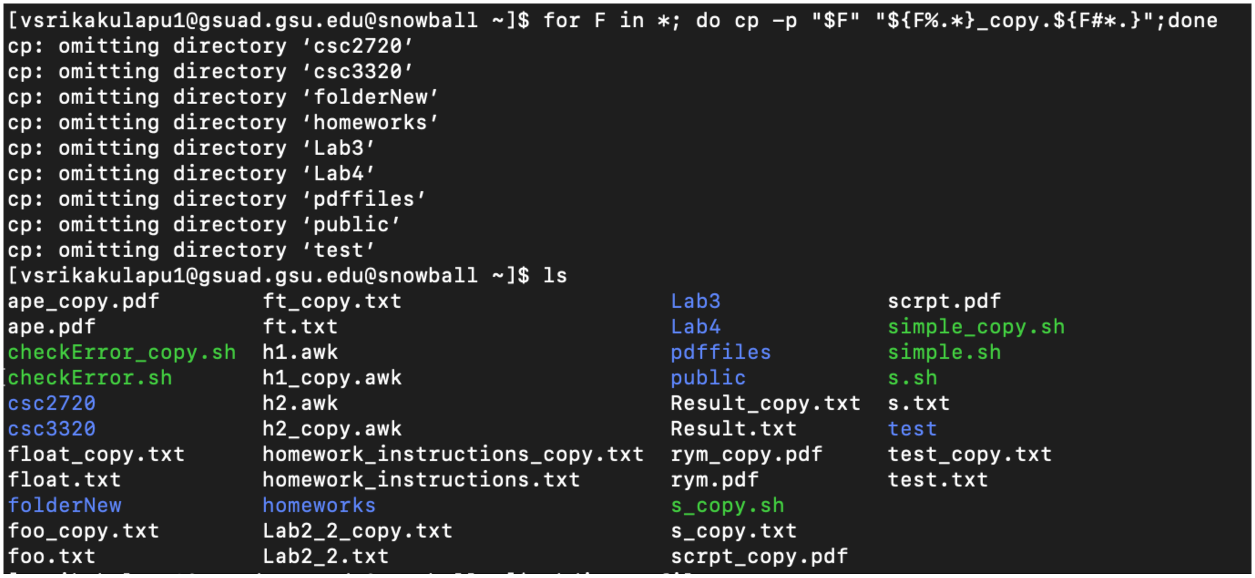
Part III Programming: 15pts

15. Sort all the files in your class working directory (or your home directory) as per the following requirements:

a. A copy of each file in that folder must be made. Append the string “_copy” to the name of the file
b. The duplicate (copied) files must be in separate directories with each directory specifying the type of the file (e.g. txt files in directory named txtfiles, pdf files in directory named pdffiles etc).
c. The files in each directory must be sorted in chronological order of months.
d. An archive file (.tar) of each directory must be made. The .tar files must be sorted by name in ascending order.
e. An archive file of all the .tar archive files must be made and be available in your home directory.

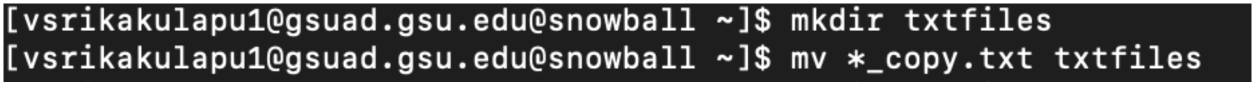
As an output, show your screen shots for each step or a single screenshot that will cover the outputs from all the steps.

a.)

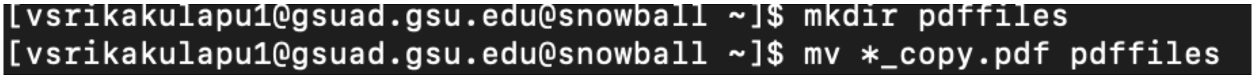


```
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ for F in *; do cp -p "$F" "${F%.*}_copy.${F#*.}";done
cp: omitting directory ‘csc2720’
cp: omitting directory ‘csc3320’
cp: omitting directory ‘folderNew’
cp: omitting directory ‘homeworks’
cp: omitting directory ‘Lab3’
cp: omitting directory ‘Lab4’
cp: omitting directory ‘pdffiles’
cp: omitting directory ‘public’
cp: omitting directory ‘test’
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ ls
ape_copy.pdf        ft_copy.txt                     Lab3              scrpt.pdf
ape.pdf             ft.txt                          Lab4              simple_copy.sh
checkError_copy.sh  h1.awk                          pdffiles          simple.sh
checkError.sh       h1_copy.awk                     public            s.sh
csc2720             h2.awk                          Result_copy.txt   s.txt
csc3320             h2_copy.awk                     Result.txt        test
float_copy.txt      homework_instructions_copy.txt  rym_copy.pdf      test_copy.txt
float.txt           homework_instructions.txt       rym.pdf           test.txt
folderNew           homeworks                       s_copy.sh
foo_copy.txt        Lab2_2_copy.txt                 s_copy.txt
foo.txt             Lab2_2.txt                      scrpt_copy.pdf
```

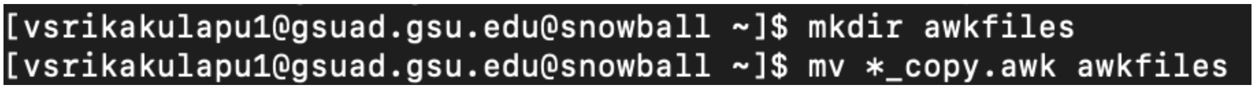
b.)



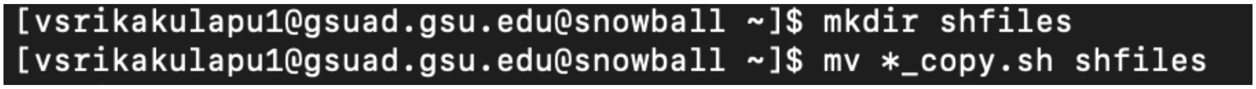
```
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ mkdir txtfiles
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ mv *_copy.txt txtfiles
```



```
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ mkdir pdffiles
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ mv *_copy.pdf pdffiles
```

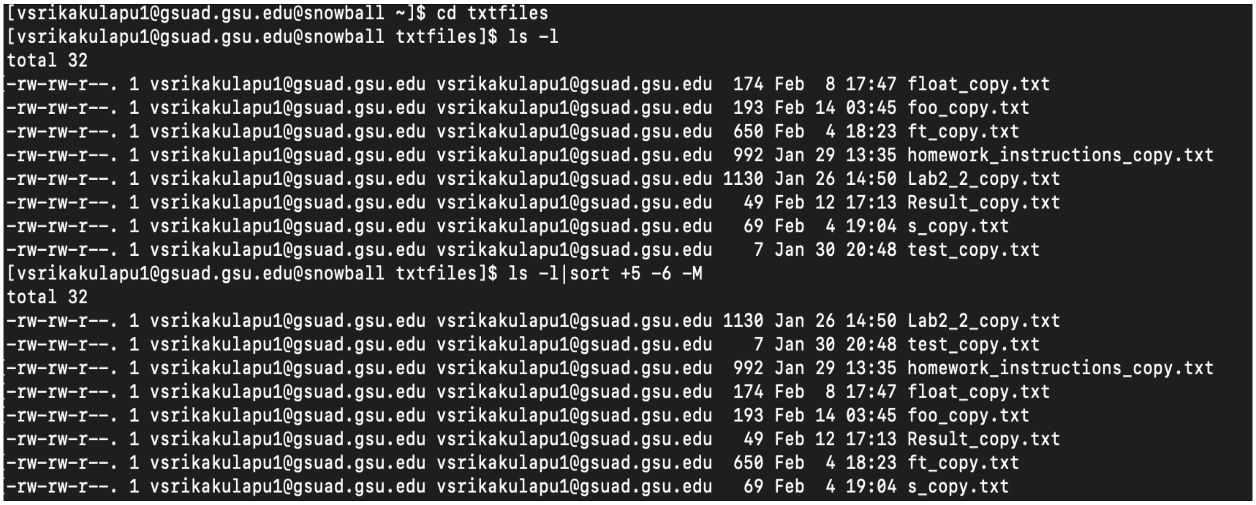


```
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ mkdir awkfiles
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ mv *_copy.awk awkfiles
```

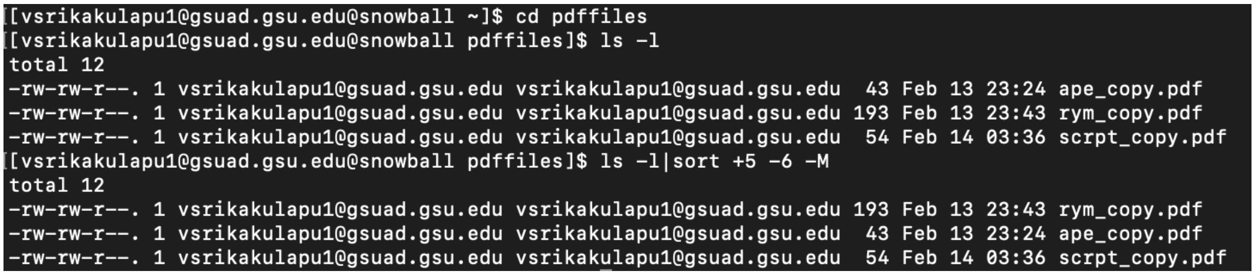


```
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ mkdir shfiles
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ mv *_copy.sh shfiles
```

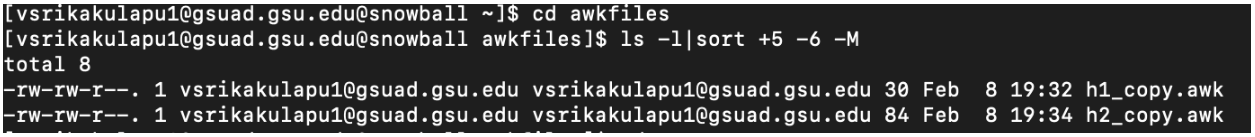
c.)



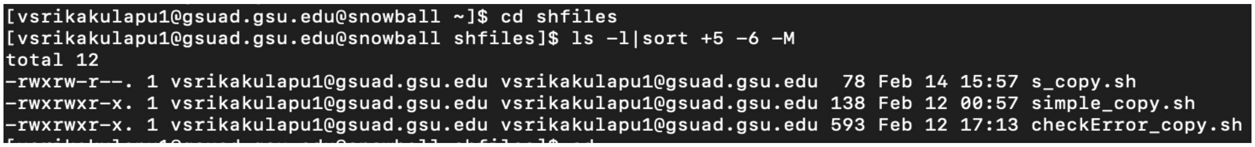
```
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ cd txtfiles
[vsrikakulapu1@gsuad.gsu.edu@snowball txtfiles]$ ls -l
total 32
-rw-rw-r--. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu  174 Feb  8 17:47 float_copy.txt
-rw-rw-r--. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu  193 Feb 14 03:45 foo_copy.txt
-rw-rw-r--. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu  650 Feb  4 18:23 ft_copy.txt
-rw-rw-r--. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu  992 Jan 29 13:35 homework_instructions_copy.txt
-rw-rw-r--. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu 1130 Jan 26 14:50 Lab2_2_copy.txt
-rw-rw-r--. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu   49 Feb 12 17:13 Result_copy.txt
-rw-rw-r--. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu   69 Feb  4 19:04 s_copy.txt
-rw-rw-r--. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu    7 Jan 30 20:48 test_copy.txt
[vsrikakulapu1@gsuad.gsu.edu@snowball txtfiles]$ ls -l|sort +5 -6 -M
total 32
-rw-rw-r--. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu 1130 Jan 26 14:50 Lab2_2_copy.txt
-rw-rw-r--. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu    7 Jan 30 20:48 test_copy.txt
-rw-rw-r--. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu  992 Jan 29 13:35 homework_instructions_copy.txt
-rw-rw-r--. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu  174 Feb  8 17:47 float_copy.txt
-rw-rw-r--. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu  193 Feb 14 03:45 foo_copy.txt
-rw-rw-r--. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu   49 Feb 12 17:13 Result_copy.txt
-rw-rw-r--. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu  650 Feb  4 18:23 ft_copy.txt
-rw-rw-r--. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu   69 Feb  4 19:04 s_copy.txt
```



```
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ cd pdffiles
[vsrikakulapu1@gsuad.gsu.edu@snowball pdffiles]$ ls -l
total 12
-rw-rw-r--. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu  43 Feb 13 23:24 ape_copy.pdf
-rw-rw-r--. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu 193 Feb 13 23:43 rym_copy.pdf
-rw-rw-r--. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu  54 Feb 14 03:36 scrpt_copy.pdf
[vsrikakulapu1@gsuad.gsu.edu@snowball pdffiles]$ ls -l|sort +5 -6 -M
total 12
-rw-rw-r--. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu 193 Feb 13 23:43 rym_copy.pdf
-rw-rw-r--. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu  43 Feb 13 23:24 ape_copy.pdf
-rw-rw-r--. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu  54 Feb 14 03:36 scrpt_copy.pdf
```

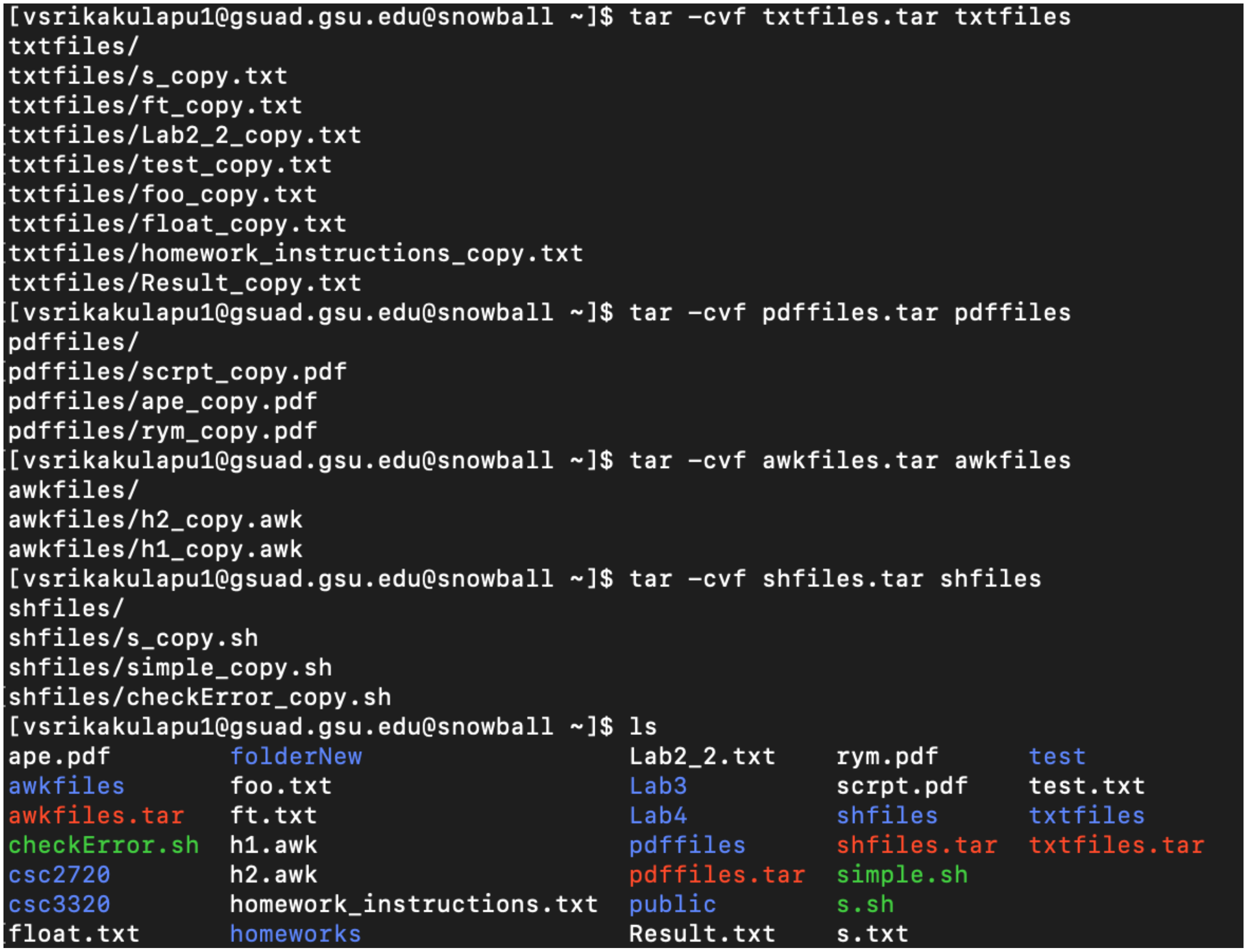


```
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ cd awkfiles
[vsrikakulapu1@gsuad.gsu.edu@snowball awkfiles]$ ls -l|sort +5 -6 -M
total 8
-rw-rw-r--. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu 30 Feb  8 19:32 h1_copy.awk
-rw-rw-r--. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu 84 Feb  8 19:34 h2_copy.awk
```

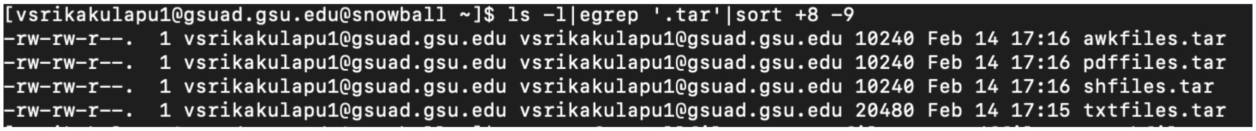


```
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ cd shfiles
[vsrikakulapu1@gsuad.gsu.edu@snowball shfiles]$ ls -l|sort +5 -6 -M
total 12
-rwxrw-r--. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu  78 Feb 14 15:57 s_copy.sh
-rwxrwxr-x. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu 138 Feb 12 00:57 simple_copy.sh
-rwxrwxr-x. 1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu 593 Feb 12 17:13 checkError_copy.sh
```

d.)

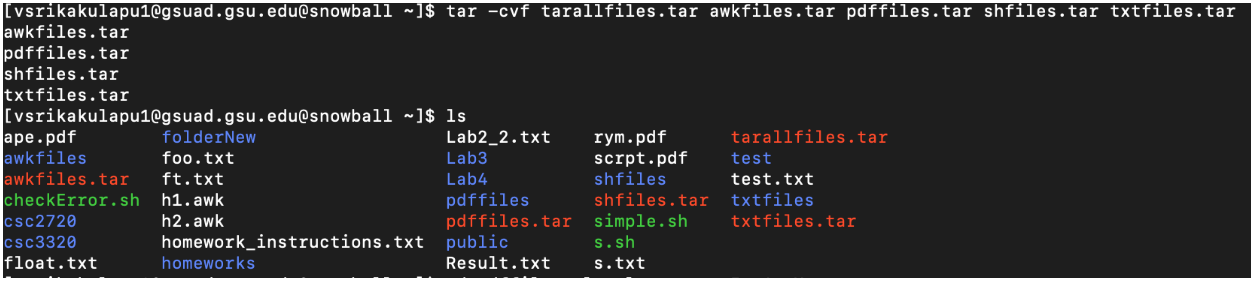


```
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ tar -cvf txtfiles.tar txtfiles
txtfiles/
txtfiles/s_copy.txt
txtfiles/ft_copy.txt
txtfiles/Lab2_2_copy.txt
txtfiles/test_copy.txt
txtfiles/foo_copy.txt
txtfiles/float_copy.txt
txtfiles/homework_instructions_copy.txt
txtfiles/Result_copy.txt
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ tar -cvf pdffiles.tar pdffiles
pdffiles/
pdffiles/scrpt_copy.pdf
pdffiles/ape_copy.pdf
pdffiles/rym_copy.pdf
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ tar -cvf awkfiles.tar awkfiles
awkfiles/
awkfiles/h2_copy.awk
awkfiles/h1_copy.awk
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ tar -cvf shfiles.tar shfiles
shfiles/
shfiles/s_copy.sh
shfiles/simple_copy.sh
shfiles/checkError_copy.sh
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ ls
ape.pdf         folderNew                  Lab2_2.txt    rym.pdf       test
awkfiles        foo.txt                    Lab3          scrpt.pdf     test.txt
awkfiles.tar    ft.txt                     Lab4          shfiles       txtfiles
checkError.sh   h1.awk                     pdffiles      shfiles.tar   txtfiles.tar
csc2720         h2.awk                     pdffiles.tar  simple.sh
csc3320         homework_instructions.txt  public        s.sh
float.txt       homeworks                  Result.txt    s.txt
```



```
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ ls -l|egrep '.tar'|sort +8 -9
-rw-rw-r--.  1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu 10240 Feb 14 17:16 awkfiles.tar
-rw-rw-r--.  1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu 10240 Feb 14 17:16 pdffiles.tar
-rw-rw-r--.  1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu 10240 Feb 14 17:16 shfiles.tar
-rw-rw-r--.  1 vsrikakulapu1@gsuad.gsu.edu vsrikakulapu1@gsuad.gsu.edu 20480 Feb 14 17:15 txtfiles.tar
```

e.)



```
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ tar -cvf tarallfiles.tar awkfiles.tar pdffiles.tar shfiles.tar txtfiles.tar
awkfiles.tar
pdffiles.tar
shfiles.tar
txtfiles.tar
[vsrikakulapu1@gsuad.gsu.edu@snowball ~]$ ls
ape.pdf        folderNew                  Lab2_2.txt    rym.pdf      tarallfiles.tar
awkfiles       foo.txt                    Lab3          scrpt.pdf    test
awkfiles.tar   ft.txt                     Lab4          shfiles      test.txt
checkError.sh  h1.awk                     pdffiles      shfiles.tar  txtfiles
csc2720        h2.awk                     pdffiles.tar  simple.sh    txtfiles.tar
csc3320        homework_instructions.txt  public        s.sh
float.txt      homeworks                  Result.txt    s.txt
```