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: Bronson Tharpe

Campus ID: btharpe2

Panther #: 002472907

Part 1:

1. Grep is used to filter files based on basic regular expressions (e.g. searching for “happy” in a file. Egrep is similar but used for extended regular expressions (e.g. searching for an expression that contains meta characters such as ‘+’ or ‘?’). Fgrep performs similarly, except it is used for fixed strings and treats every character literally (e.g. searching for ‘.’ in a file is easier when using an fgrep, because you don’t have to precede it with an escape character).
2. The tar utility is used to compress and decompress files. Tar can also compress multiple files into a single file by using the -cvf exception (e.g. tar -cvf combined_files.tar first_file second_file).
3. The awk and sed utilities can be used to break the lines into fields by separators and remove selected lines, respectively. The default separator is a single space, but it can be changed by changing the FS variable, e.g. awk ‘BEGIN { FS = “&” };’
4. The sort command sorts a file in ascending or descending order, line by line. The possible fields are -r to sort in descending order, -t to specify a separator, -b to ignore leading blanks, -f to ignore case, and -M to match month, -u to remove duplicates, -c to check if it’s already been sorted, -k to sort a table by column number, -n to sort numerically, and -nr to sort in reverse numeric order. E.g. sort -bfr would sort in descending order ,ignoring leading blanks and case.

Part 2a:

5. Hello World!!!
6. Awk script outputs:
    a. 1 <= NF { print $5 }
        i. Prints the 5th field of each line.
    b. NR >= 1 && NR >= 5 { print $1 }
        i. Prints the 1st field of lines 1...5 of the file.
    c. 1,5 { print $0 }
        i. Prints every field of each line.
    d. {print $1 }
        i. Prints the first field of each line.
7. good
8. /\+$/{ print $0}
9. Deleting lines:
    a. Delete the first 5 lines: sed -i '1,5d' foo
    b. Delete the last 5 lines: head -n -5 foo

Part 2b:

10. Describing the function and outputs:
    a. cat float
        i. Prints the contents of the file.
    b. cat h1.awk
        i. Prints the contents of the awk script.
    c. NR>2 && NR<4{print NR ":" $0
        i. This is a non-functional awk script. It is missing the closing brace.
    d. awk '/.*ing/ {print NR ":" $1}' float
        i. This command executes an awk script that prints the row number along with the first item of a matching line that meets the search pattern criteria of any character preceding "ing" (1:Wish <break> 3:When <break> 4:Now).
11. Describing the function and outputs:
    a. awk -f h1.awk float
        i. This command executes h1.awk on the file float and prints the third line of the file preceded by the line number and a colon (3:When everything seemed so clear.).
12. Describing the function and outputs:
    a. cat h2.awk
        i. Prints the contents of the awk script.
    b. awk -f h2.awk float
        i. Executes h2.awk on float and prints the first and last word of each line, separated by a comma, sandwiched between "Start to scan file" and "END".
13. Describing the function and outputs:
    a. sed 's/\s/\t/g' float
        i. This function replaces all instances of 's' in float with tabs.
14. Describing the function and outputs:
    a. $ ls *.awk| awk '{print "grep --color 'BEGIN' " $1 }' |sh
        i. This function lists all the awk files, pipes this into an awk function that uses grep to search the files for the word "BEGIN" and print it in color along with everything else on it's line (this is all piped to sh to be executed as a shell script.
15. Describing the function and outputs:
    a. mkdir test test/test1 test/test2
        i. This command creates a directory "test" and puts 2 new subdirectories inside of it (test1 and test2).
    b. cat>test/testt.txt

i. This command creates a file called “testt.txt” and allows the user to write the contents of the file (This is a test file) until <ctrl d> is pressed.

c. cd test
   i. This command changes the working directory to test.

d. ls -l . | grep '^d' | awk '{print "cp-r" $NF "" $NF ".bak"}' | sh
   i. This command lists the permissions of every file in the current directory, pipes the result into a grep command that searches for “d” at the beginning of the line, pipes the result of this into an awk script that prints “cp -r” followed by a space, the last field from the line, another space, the last field from the line plus “.bak”. All of this is then piped to run as a shell command, which will run the outputs one at a time.
   ii. This command shouldn’t give any text output in the terminal, but will create copies (with an added “.bak” for directories) of each directory in addition to its content that resides in the current working directory.

Part III Programming:

1. Make a directory for txt files:

```
[btharpe2@gsuad.gsu.edu@snowball ~]$ mkdir txtfiles
```

2. Find all txt files and put them in this new directory.

```
[btharpe2@gsuad.gsu.edu@snowball ~]$ find . -name '*.txt' -exec cp -prv '{}' '/home/btharpe2/txtfiles/' ';'
```

```
[btharpe2@gsuad.gsu.edu@snowball ~]$ ls txtfiles
CSC_Course.txt            mountainList.txt  newList.txt
homework_instructions.txt  myLab2.txt       test.txt
```

3. Make a directory for csv files:

```
[btharpe2@gsuad.gsu.edu@snowball ~]$ mkdir csvfiles
```

4. Find all csv files and put them in this new directory:

```
[btharpe2@gsuad.gsu.edu@snowball ~]$ find . -name '*.csv' -exec cp -prv '{}' '/home/btharpe2/csvfiles/' ';'
```

```
[btharpe2@gsuad.gsu.edu@snowball ~]$ ls csvfiles
myRealEstate.csv  RealEstate.csv
```

5. Rename contents of csvfiles and txtfiles by appending "_copy" to their existing name.

```
[btharpe2@gsuad.gsu.edu@snowball ~]$ cd txtfiles
[btharpe2@gsuad.gsu.edu@snowball txtfiles]$ ls -l . | awk '{print "mv " $NF " " $NF "_copy"}' | sh
```

```
[btharpe2@gsuad.gsu.edu@snowball txtfiles]$ ls
CSC_Course.txt_copy  homework_instructions.txt_copy  mountainList.txt_copy  myLab2.txt_copy  newList.txt_copy  test.txt_copy
[btharpe2@gsuad.gsu.edu@snowball txtfiles]$
```

```
[btharpe2@gsuad.gsu.edu@snowball ~]$ cd csvfiles
[btharpe2@gsuad.gsu.edu@snowball csvfiles]$ ls -l . | awk '{print "mv " $NF " " $NF "_copy"}' | sh
```

```
[btharpe2@gsuad.gsu.edu@snowball csvfiles]$ ls
myRealEstate.csv_copy  RealEstate.csv_copy
```

6. Sort files in both of the copy directories by chronological order of months.

```
[btharpe2@gsuad.gsu.edu@snowball csvfiles]$ ls -l . | awk '{print "sort -M " $NF}' | sh
```

```
[btharpe2@gsuad.gsu.edu@snowball txtfiles]$ ls -l . | awk '{print "sort -M " $NF}' | sh
```

7. Make archive files, sorted in ascending order, for both directories.

```
[btharpe2@gsuad.gsu.edu@snowball ~]$ tar -cvf --sort=name txtarchive.tar /home/btharpe2/txtfiles
```

```
[btharpe2@gsuad.gsu.edu@snowball ~]$ tar -cvf --sort=name csvarchive.tar /home/btharpe2/csvfiles
csvarchive.tar
```

```
[btharpe2@gsuad.gsu.edu@snowball ~]$ ls
csc3320  csvarchive.tar  csvfiles  homeworks  Lab3  Lab4  public  simple.sh  --sort=name  txtarchive.tar  txtfiles
```

8. Make an archive file of all .tar archive files (and store it in my home directory):

```
[btharpe2@gsuad.gsu.edu@snowball ~]$ tar -cvf tararchive.tar csvarchive.tar txtarchive.tar
csvarchive.tar
txtarchive.tar
[btharpe2@gsuad.gsu.edu@snowball ~]$ ls
csc3320  csvarchive.tar  csvfiles  homeworks  Lab3  Lab4  public  simple.sh  --sort=name  tararchive.tar  txtarchive.tar  txtfiles
[btharpe2@gsuad.gsu.edu@snowball ~]$
```