Exercises-Filters1

The files referred to in this exercise set are on the **`samples/Data`** directory in the public data area.

Part One

Description

In this part we will practice with the Unix filters **`head`**, **`tail`**, **`cat`**, and **`fold`**. We will also practice using **`wc`**, even though it is not a filter. We will use other Unix commands to get raw data for our filters.

`head`, **`tail`** and **`cat`** are used to display lines from a file. **`cat`** can also add line numbers. **`wc`** is used to count the numbers of words, lines and characters in a file. Review their use before proceeding.

Note on numbering lines: lines can be numbered using **`cat -n`**

Exercises

For this part, you will be using the file **`jbpart.faq`** unless otherwise indicated.

1. Using **`cat`**, display **`jbpart.faq`** with line numbers.
2. You may have noticed that some of the lines wrapped. Use **`fold`** with some options so that the resulting lines are broken on a space. (Use a width of 60 characters) Are the broken lines numbered? (You may have to look at the *Hints* for the answer here, since this may not be covered in class.)
3. Redo the last command so that the broken lines are numbered.
4. Output how many lines there are in **`jbpart.faq`**. Do this so that you only get the number of lines, not the name of the file.
5. Output the number of characters in **`jbpart.faq`**. Compare this to the size in an **`ls -l`** listing.
6. Output the first 10 lines. Redo the command, adding line numbers.
7. Output the last 10 lines. Redo the command, adding the correct line numbers from the original file.
8. Output lines **`15-25`** with the correct line numbers.
9. Using the **`groups`** command, output the number of groups you are a member of.

The rest of these commands assume you are in the **`Data`** directory

10. Count the number of objects in the **`Data`** directory, not including hidden objects.
11. Count the number of objects beneath (recursively) the **`samples`** directory (one directory up), including hidden objects.
12. Count the number of regular files beneath the **`samples`** directory, including hidden files.
13. How many regular files are there beneath the public data area whose name ends in **`.txt`**

Part Two

Description

In this part will will add practice with **`grep`** using the options **`-i`** and **`-v`** and anchors, and **`sort`** with the options **`-n`** and **`-r`**

Procedure

Using the file **`itsann.txt`** in the **`samples/Data`** directory, do the following

1. output the lines that contain **`ccsf`**
2. broaden the search to include lines that contain **`ccsf`** in any mixture of upper- and lower- case characters.
3. output the number of lines found in the last problem.
4. output the lines with **`ccsf`** (case-insensitive) with the original line numbers from the file.
5. output the number of lines that do not contain **`ccsf`** (case-insensitive)
6. output the lines that begin with an upper-case **`T`**. How many are there?

7. output the lines that end with an **`e`** in either case. How many are there?

Using the file **`numbers`**, do the following:

8. sort the file alphabetically
9. sort the file numerically
10. output the largest eight numbers from the file **`numbers`**
11. list the name and size (in characters) of each object in the current directory
12. list the name and size of each object in the current directory whose name starts with a lowercase letter.
13. output the five largest objects in the current directory whose name starts with a lowercase letter.

Part Three - sort and uniq

Description

`sort` has many options. Unfortunately, although the options have meaning similar to other commands, the options themselves are different. This is a common source of confusion. Review these options carefully before doing these exercises. The most confusing differences are:

The option to ignore case in sort is **`-f`** (fold case). With most tools it is **`-i`**. No matter, on current linux systems that use the standard 'locale', upper- and lower-case is folded by default (**`-f`** is on by default).

The option to specify the delimiter in sort is **`-t`** (tab character). With most tools it is **`-d`**

The field numbers and how to specify sort keys. There are two ways: the old way and the new (POSIX) way. We will cover the POSIX way, but you should know about the old way since it is so widely used. The most confusing difference is that field numbers in the old way (+field -field) are numbered starting with 0. In the POSIX (-kfield1,field2) they are numbered starting with 1

Example: **`sort -k1,3`** (POSIX) is **`sort +0 -2`** (old way) # sort on fields 1, 2 and 3

Exercises

Look at the file **`st`**. Using this file, output the following using **`st`**

1. sorted alphabetically on the first field
2. sorted numerically on the first field
3. sorted alphabetically on the second field
4. sorted alphabetically on the second word of the second field
5. sorted numerically by the numeric portion of the third field (you will have to figure this out looking at the sytax for sort in the Filters handout.)
6. sorted by the third field then by the first field.

Examine **`st`** again. Look for duplicate records. There are two sets of two duplicates each.

7. Run **`st`** through **`uniq`**. Is either duplicate removed?
8. Do what is needed to remove both duplicates from the output.

`sort` has its own 'unique' option, **`-u`**, but this means 'remove records with duplicate keys'. Output **`st`** after removing:

9. records with duplicate last fields
10. records with duplicate field #3s
11. Output records from **`st`** sorted by email (field #4) after removing records with duplicate field #2s. Treat upper and lower case the same.
12. When is the **`-u`** option of **`sort`** the same as **`uniq`**?

Continue with the following sort problems on the data source indicated

13. List the samples/Data directory sorted by file size (i.e., an ls -l listing sorted by size)

14. Output the Emp_Manager file sorted first by the job field (the last field) then by the login field (the next-to-last field). Ignore case in the job field.
15. Repeat the last sort, but remove records that have duplicate field #2's
16. Output the passwd file sorted first by field 4 then by field 1

Answers

Part One

1. **`cat -n jbpart.faq`**
2. **`cat -n jbpart.faq | fold -s -w60`** # the default width is 80
3. **`fold -s -w 60 jbpart.faq | cat -n`** # you should really decrease the width to account for the line numbers
4. **`wc -l jbpart.faq`** # outputs the name of the file, too
   `wc -l < jbpart.faq` # now wc doesn't know the name of the file
5. **`wc -c < jbpart.faq`**
6. **`head -n 10 jbpart.faq`**
   `head -n 10 jbpart.faq | cat -n`
7. **`tail -n 10 jbpart.faq`**
   `cat -n jbpart.faq | tail -n 10`
8. **`cat -n jbpart.faq | head -n 25 | tail -n 11`**
9. **`groups | wc -w`**
10. **`ls | wc -l`**
11. **`find .. | wc -l`**
12. **`find .. -type f | wc -l`**
13. **`find ../.. -type f -name "*.txt" | wc -l`**

Part Two

1. **`grep 'ccsf' itsann.txt`**
2. **`grep -i 'ccsf' itsann.txt`**
3. **`grep -i 'ccsf' itsann.txt | wc -l`**
4. **`cat -n itsann.txt | grep -i 'ccsf'`**
5. **`grep -vi 'ccsf' itsann.txt | wc -l`**
6. **`grep '^T' itsann.txt`** (then pipe it to **`wc -l`** to count them)
7. **`grep -i 'e$' itsann.txt`** (then pipe it to **`wc -l`** to count them)
8. **`sort numbers`**
9. **`sort -n numbers`**
10. **`sort -n -r numbers | head -n 8`**
11. **`wc -c *`**
12. **`wc -c [[:lower:]]*`**
13. **`wc -c [[:lower:]]* | sort -n -r | head -n 6 | tail -n 5`** (this gets rid of the annoying **`totals`** line)

Part Three

1. **`sort st`**
2. **`sort -n st`**

3. **`sort -t# -k2,2 st`**
4. **`sort -t' ' -k2,2 st`**
5. **`sort -t# -k3.2,3n st`**
6. **`sort -t# -k3,3 -k1,1n st`**
7. **`cat st | uniq`** or **`uniq < st`** # the adjacent duplicates are removed
8. You must sort the file so that all duplicates are adjacent:
   `sort st | uniq`
9. **`sort -t# -k5,5 -u st`**
10. **`sort -t# -k3,3 -u st`**
11. **`sort -t# -k2,2f -u st | sort -t# -k4,4f`** # you could put **`-f`** before the **`-k`** instead
12. They are the same when working on the entire record: **`sort | uniq`** is the same as **`sort -u`**
13. **`ls -l | sort -k5,5n`**
14. **`sort -t: -k6.6f -k5,5 Emp_Manager`**
15. **`sort -t: -k2,2 -u Emp_Manager | sort -t: -k6,6f -k5,5`**