University of Colorado Denver

CSCI 4580/5580 DATA SCIENCE
Spring 2022

Lab 1: Introduction to Command Line Utilities for Data Science

Deadline: February 4, 2022 – 11:59 PM

Total Points: 100

In this lab you will be using Unix command line utilities to do some data cleaning and basic analysis. Please note that this lab must be done individually. By submitting this lab, you certify that that this is your own work, your code will be checked against other submissions and resources using automatic tools.

Instructions

- You will be using the Ubuntu virtual machine that you created in lab 0 to complete this lab. If you didn't complete lab 0, please refer to the instructions there to install the required tools before continuing with this lab.
- This lab uses datasets downloaded from Canvas. You will need a network connection in your virtual machine. Make sure your computer has a working connection before you start the virtual machine.
- You can view the manual pages of all commands used in this lab by typing man <command-name> in your terminal. For example, man ls will produce the documentation of the list (ls) command.

Submission

You need to submit a single PDF file on Canvas, named *your-lastname_your-first-name.pdf*. For example, if your name is John Smith, you should name the file *smith_john.pdf*.

- Your submission should only contain your answers to Questions 1-12. Please do not include extra files such as the input datasets in your submission.
- Please download your submission file after submission and make sure it is not corrupted. We will not be responsible for corrupted submissions and will not take a resubmission after the deadline.

Need Help?

If you need help with this lab, please email me at sundous.hussein@ucdenver.edu or come to my office hours. We also encourage you to ask your questions on the designated channel for the lab on Microsoft Teams. This way, you may receive assistance from your classmates that might've ran through the same issues.

Section 1: File System Analytics

In this section, we'll use some of the basic Linux commands to do some analysis of the file system on your Ubuntu virtual machine.

Navigating the File System

If you've used the terminal on a Linux system before, you're probably familiar with the ls and cd commands. ls has some great extended options but making use of the results of ls can be difficult and tasks like getting access to filenames recursively can be tricky.

The find Command

find provides a much more flexible interface to listing files that match a particular criterion than ls. You can easily restrict yourself to a particular type of file (e.g., directories or plain files), recurse indefinitely or to some maximum depth, dispatch an action of filenames than match your search, or pair the results of find with a command like xargs to further manipulate files that match your filters.

For example, the command find /usr/bin -type f -name 'py*' will produce the list of names of files (not directories) in /usr/bin that start with 'py'.

The du Command

The du command allows you to report usage statistics of files and directories in your system. By default, it reports its usage statistics at the block level, and you can use the -h option to display the information in a human-readable format. It's worth mentioning that the results of du are not exact but rather sizes are reported as multiples of block size.

Try running du -hs * in your home directory to see how much space your files and directories are consuming.

Sorting

The sort command is a powerful command. In most systems, it is implemented via an external merge sort which makes it possible to sort huge files, even ones that don't fit into memory, in a reasonable amount of time.

By default, sort will sort items lexicographically but with the -n flag, it will sort in numeric order. This means that 10100 will come after 2 in numeric sort order even though it precedes 2 lexicographically. In recent versions of GNU sort, the command also takes a -h parameter, which allows you to sort human-readable numbers like file sizes produced by du -h.

Additionally, sort will let you select exactly which fields of a file you want to sort on (by default, it's the whole line), and in which order these fields should be sorted.

Exercises

Download this Python repository: https://github.com/tensorflow/models/archive/master.zip.

Unzip the repository and then open the terminal in the root folder of the repository or open the terminal and use the cd command to get to this directory:

1. Find all text files (files with extension .txt) under models-master directory. Run your command and take a screenshot from the terminal (your command should be included in the screenshot as well as the top 10 files in the output). [5 points]
2. Find all files under models-master/official/ that do not end with .py. [5 points]
3. Find all files with group write permission under models-master/official/nlp. (Hint: review the -perm option of the find command). [5 points]
4. Find all files ending with '.txt' or '.json' under models-master. [5 points]
5. Modify your command from exercise 1 to sort the list of text files lexicographically. (Hint: use the pipe | operator and pass the results of the find command to sort. [10 points]
6. Inspect which files and directories are using most of the space in models-master, i.e., sort all files and directories under models-master based on their size in reverse order. (Hint: use the du command). [10 points]

Section 2: Log Files Processing with Command Line Tools

In this section we will examine tools that you can use to analyze and explore text files quickly.

Downloading the Data Files

The first step in data analysis is typically acquiring the data files that you need to process.

HTTP Logs Dataset

We will be using HTTP access logs from the 1998 soccer World Cup website. The dataset (wc_day6_1.log) is available on Canvas under Lab 1.

Examine the file by running less wc_day6_1.log. This will show you the first few lines of the file and you can page through the file using the arrow keys. You will notice that each hit or access to the website is logged in a new line in the log file. The format of each line is in the Common Log File Format and this format is supported by most HTTP servers. In this case, the data has been anonymized. Let's look at one line from the file to explain each field:

57 - - [30/Apr/1998:22:00:48 +0000] "GET /english/images/lateb_new.gif HTTP/1.0" 200 1431

In the entry above, 57 refers to the ClientID, a unique integer identifier for the client that issued the request. While this field typically contains the IP address, for privacy reasons it has been replaced with an integer. The next two fields are – to signify that the fields are missing from the log entry. Again, these correspond to user-identifier and userid and have been removed for privacy reasons.

The next field is the time at which the request was made and is followed by the HTTP request. In this example a GET request was made for lateb_new.gif. The next field is the HTTP return code, in this example, it is 200. You can find a list of codes and their meanings here. The last field is the size of the object returned to the client, measured in bytes.

Exploring the Dataset

Before we get to the exercises, let's explore the dataset and try out some basic commands. First, let's count how many visits the website got. To do this we just count the number of lines in the file by running wc -l wc_day6_1.log. Your output should be 46212 wc_day6_1.log.

We can do something more interesting by finding out how many times the ticketing webpage was visited. To do this you could run grep tickets wc_day6_1.liog | wc -l which would give you 1193. However, the previous grep command counts images and other elements that have the word tickets in their path. To restrict our search to html pages, we can use a regular expression with the grep command grep "tickets.*html" wc_day6_1.log | wc -l which should give you 100.

We can also prune the dataset to only look at interesting parts of it. For example, we can just look at the first 50 URIs and their sizes using the head and cut commands:

```
head -50 wc_day6_1.log | cut -d ' ' -f 1,7
```

In the above command, the -d flag denotes what delimiter to use and -f states what fields should be selected from the line. Try out different delimiter and field values to see how cut works.

Finally, we can see how many unique URIs are there in the first 50 visits. To do this, we could run something like head -50 wc_day6_1.log | cut -d ' ' -f 7 | sort | uniq | wc -l. Here we use the tool uniq to only count unique URIs. Note that the input to uniq should be sorted, so we use sort before calling uniq.

The uniq tool can also be used to count how many times an item occurs by passing it the -c flag. For example, if we run the same command as above but with uniq -c we'll get

```
head -50 wc_day6_1.log | cut -d ' ' -f 7 | sort | uniq -c | tail -10
```

```
1 /images/home_fr_phrase.gif
2 /images/home_intro.anim.gif
1 /images/home_logo.gif
1 /images/home_sponsor.gif
1 /images/home_tool.gif
1 /images/logo_cfo.gif
1 /images/nav_bg_top.gif
```

1 /images/team_hm_afc.gif

1 /images/team_hm_caf.gif

1 /images/team_hm_concacaf.gif

This shows that /images/home_intro.anim.gif occurred twice in the first 50 URIs.

Exercises

Now use the above tools to answer some analysis questions:

7. How many URIs have return codes of 200? [10 points]
8. What are the 5 most frequently visited URIs (consider only the .html pages)? Your command must only output the top 5 URIs and their counts. Provide a screenshot that includes your command and its output. [10 points]
9. Filter all entries (requests) made on 30th April 1998 at 22:53. [10 points]

Section 3: Data Transformation with sed

In this section, we will use regular expressions to manipulate date. We've prepared a quick refresher of regular expressions in the accompanying supplementary document of this lab.

Regular Expressions Substitution in sed

sed (short for "Stream EDitor") is a tool for modifying and extracting data from text files. The most common sed command is regular expressions substitution.

The most notable flag of the sed command is g, which causes sed to find and replace all instances of the pattern rather than the first one.

For example,

echo "The quick brown fox jumps over the lazy dog." | sed 's/[tT]he [a-z]*/The yellow/'

matches only 'The quick', resulting in 'The yellow brown fox jumps over the lazy dog.', while

echo "The quick brown fox jumps over the lazy dog." | sed 's/[tT]he [a-z]*/The yellow/g'

matches both 'The quick' and 'the lazy' and results in 'The yellow brown fox jumps over The yellow dog.'

You can always run man sed to review details of the flags of the command.

Removing Cruft in sed

Let's start by removing the cruft from dates in the log file. Each date is surrounded by a [and +0000]. We can remove the +0000] by using sed 's/ +0000]//'.

To try this on the first ten lines of the log file, use:

head wc_day6_1.log | sed 's/ +0000]//'

Similarly, we can remove [using s/\[//. We need the backslash before the [to prevent sed from treating it as a special character in its regular expression syntax. We can combine this our previous cleaning using:

head wc_day6_1.log | sed 's/ +0000]//' | sed 's/\[//' or, only invoking sed once head wc_day6_1.log | sed 's/ +0000]//; s/\[//'.

Back References in sed

sed supports back references in substitutions, allowing you to include part of the matched text in the replacement string. To use back references, make a capturing group by surrounding part of the matched text with \(and \); then in the replacement text, use \1 to place the text of the first group, \2 for the second and so on.

For example:

echo "The quick brown fox jumps over the lazy dog." | sed 's/\([Tt]he\) \([a-z]*\)/\1 "\2"/g' results in The "quick" brown fox jumps over the "lazy" dog.

Exercises

10. Write a command that converts the "The quick brown fox jumps over the lazy dog." To "quick The brown fox jumps over lazy the dog.". [10 points]
11. In the following sentence add [] around words starting with s and containing e and t in any order "subtle exhibit asset sets tests site". [10 points]
12. Use the sed command to change the following date format from 13/April/2007 to 2007-April-13. (Hint: use the echo command to print the date and then pipe the output to the sed command). [10 points]