Operating Systems Laboratory (CS39002)

Spring Semester 2021-2022

Assignment 1: Familiarization with shell scripting and shell commands

Assignment given on: January 11, 2022

Assignment deadline: January 21, 2022, 11:55 PM

#READ THE FULL INSTRUCTIONS VERY CAREFULLY FIRST

0. Overview

In this assignment we will start with a simple exercise – familiarization with shell scripting. Shell or "bash" (a variant of shell software) is your interface to access a multitude of software (often called "commands"), some of which are directly provided by the OS. Shell OR command line OR terminal can be simply accessed in your Linux distribution by opening the terminal app. Search Google to know more.

0.1. Shell scripting example

Let's take an example: You have a file named "tmp.txt" which contains multiple lines, each line is a number. You want to numerically sort those numbers. So, you can simply

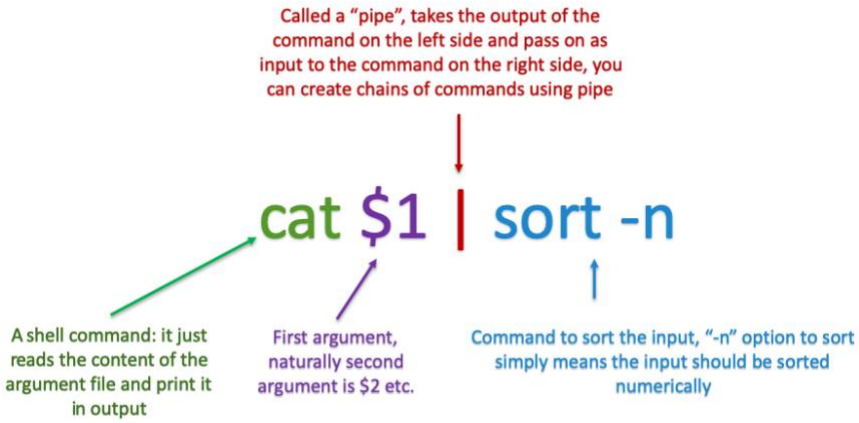
- Open up the terminal.
- Type cat tmp.txt | sort -n and press Enter.
- The sorted number will be printed in the terminal

You can also store it in a file for permanent use.

- You open a file called "sort.sh" where you would store your bash script.
- You want sort.sh to take filename as input
- So, within sort.sh you would put cat $1 | sort -n, save the file and run the following command: sh sort.sh tmp.txt
- sort.sh is called a shell script

0.2. What is going on?

Here is a dissection of the script:



Of course, like any self-respecting programming language you can do everything in shell script (loops, conditionals, case-switch etc.). However, the power of the shell script is derived from clever use of the already-existing software and functionalities provides by shell script (e.g., cat, sort, | etc.)

0.3. Shell commands to be familiar with

There are a few shell commands that you should be familiar with to efficiently use shell (aside from the shell scripting syntax). Search google (also write man <commandname> in terminal) to know more about them (some or all of them might also be useful for you to solve this assignment). Each command can take multiple arguments to perform various tasks.

- cat
- sort
- awk (also search awk one-liners)
- head
- tail
- cd
- mkdir
- ls
- touch
- file
- tr
- grep
- alias
- dd
- nohup
- wc

- split
- cut
- sed
- curl
- cmd1 &
- cmd1 | cmd2
- file1.txt < file2.txt
- file1.txt > file2.txt

0.4. Grading

- You need to submit your assignment to Moodle within the deadline mentioned.
- Make sure the submission naming scheme exactly follows as given in the assignment. Naturally if you mistype your group number, your marks might be awarded to another group.
- Submit ONE assignment per group.
- We will award 90% marks for this assignment if your solutions to the problems are correct and you can demo the correctness to your instructor in the lab with instructor mentioned test cases (means your code runs in *reasonable* time for those cases).
- The remaining 10% will be reserved for the top 5 groups who can write the code (in total) using the least number of words (akin to solving a maths problem in the least number of steps). Here "words" will mean the number of words calculated by the "wc -w" command on the terminal.
- We will release a leaderboard for this assignment.

With this introduction let's start the problems. Best of Luck!

1. Problems

Write programs in shell script under the Linux environment that would run in bash terminal and perform the following Tasks.

1.1. Computation - Prime Factorisation (10 marks)

- Write a shell script which will take an integer as a command line argument and output its prime factors as "a b c d" in ascending order including repetitions.
- Name your shell script "Assgn1_1a_< groupno>_<roll no. 1>_< roll no. 2>.sh"
- **Example input**: Assgn1_1a_< groupno>_<roll no. 1>_< roll no. 2>.sh 280
- **Example output**: 2 2 2 5 7

1.b. File Manipulation (10 marks)

- You are given a folder with hundreds of text files (download from here and unzip): https://cse.iitkgp.ac.in/~mainack/OS/assignments/2021/01/1.b.files.zip
- You need to sort each of these individual files in increasing order (numerically), write the sorted files in a new "1.b.files.out" folder (keep the name same), and then merge these individual sorted files into a single file (name it "1.b.out.txt") containing the integers and their frequency across all the text files. The integers should be sorted in increasing order (each line is of the form <num> <freq>).
- Name your shell script "Assgn1_1b_< groupno>_<roll no. 1>_< roll no. 2>.sh"

1.c. File/Directory Handling (15 marks)

- You have a directory with numerous subdirectories (which in turn could have more subdirectories within them) and files (download from data and unzip)
- Write a shell script which will iterate through all the files in the directory and segregate them based on their extensions. The script should create folders (the name of each folder being an extension, eg: txt, doc, ppt, etc.) in the root directory (1st level of the directory) and move all the files having a particular extension to the corresponding folder. For instance, all .txt files should be stored inside a folder "txt". Remove all other folders from the directory.
- Note: Some files might not have any extension. Put those files under the 'Nil' folder.
- Name your shell script "Assgn1_1c_< groupno>_<roll no. 1>_< roll no. 2>.sh"

1.d. File Handling (10 marks)

- You are given files which have numerous lines and each line having space separated words. (download Data and unzip)
- Create a new file for each given file with the same name in a folder named 'files_mod', where each line is prepended with the line number at the start and converted to comma separated words (instead of spaces).
- Example: Say the 10th line is "This is OS laboratory" (without the quotes). The line should be changed to "10,This,is,OS,laboratory" (without the quotes). Do this for all the lines in the input file.
- Name your shell script "Assgn1_1d_< groupno>_<roll no. 1>_< roll no. 2>.sh <input_file>"

1.e. Using Curl, Environment Variable and More (30 Marks)

Curl is an extremely powerful command line tool for making network requests, i.e., downloading webpages (or as they are called online resources) from online locations. Curl also takes a lot of input arguments, often signifying that some url fetch requests might require additional information to be fetched. This additional data is called request headers (refer to the demo during class). This part of the assignment will deal with using curl with additional inputs as arguments.

- Note that request headers contain more information about the resource to be fetched, or about the client requesting the resource. Similarly, Response headers provide information about the resource provided, or about the server sending the response.
- Also note that shell commands can access user-defined variables, they are called environment variables. Check Google to know the way to set environment variables for shell.
- Now do the following in a shell script:
    - Set environment variable "REQ_HEADERS" as a string with comma-separated headers. For example, REQ_HEADERS="Connection,Keep-Alive".
    - Using Curl GET https://www.example.com/ (i.e., fetch the webpage) and save as "example.html" in the same directory.
    - Send a GET request to a url using Curl to get your IP. Print the IP and the response headers (which url to use: ref).
    - Send a GET request using Curl to http://headers.jsontest.com/. Parse the JSON response (ref), read the environment variable "REQ_HEADERS" and print the required headers from the response.
    - Download JSON files from here (data credits to https://json-generator.com/) manually. Now in your script programmatically check whether the JSON syntax of each of the downloaded files is valid (ref), and write the names of the valid and invalid JSON files in two new text files - valid.txt and invalid.txt. The file should be in ascending order in the output files.
    - Add a VERBOSE argument to print debug information that'll help debug the script. This option should be disabled by default.
    - Feel free to use online resources to find the required arguments, calls, or any other reference. The outputs for each task MUST BE in the console unless an output file is mentioned explicitly.
    - References to check for this assignment
        1. Manual (Curl)
        2. https://www.jsontest.com/
        3. An Introduction to Environment Variables and How to Use Them
        4. What is Verbose Mode
        5. Approaches to add Verbose in script
        6. Parsing JSON string in Unix
- Name your shell script "Assgn1_1e_< groupno>_<roll no. 1>_< roll no. 2>.sh"

1.f. Create a frequency distribution from a large file (10 marks)

- You have a text file with around 1.5 million lines (download from https://cse.iitkgp.ac.in/~mainack/OS/assignments/2021/01/1c_input.txt.zip and unzip)
- This text file has four space-separated columns, example of three lines from the file:

```
1 ib Jim 34
1 cr JoHn 24
1 ut MaRY 46
```

- Write a shell script which will take two inputs: (i) the input file, and (ii) a column number as a user provided input (varies from 1 to 4). For that column number the code

will read entries of that column from the input file (e.g., in case of column number 3 as input the entries will be Jim, JoHn, MaRY), and convert these entries to lower case.

- Finally the script should create a file “1c_output _<column_number>_column.freq”. This file will have two columns: first is a string (converted to lowercase) from the given column and second is the frequency of that string. Sort the “1e_output _<column_number>_column.freq”. The file should be sorted in decreasing order of the second column values (i.e., frequency).
- Name your shell script “Assgn1_1f_< groupno>_<roll no. 1>_< roll no. 2>.sh”

1.g. Pattern matching (15 marks):

- Create a csv file with 150 rows and 10 columns. Fill the csv file with **random integer data (each cell value is a random integer)**. Generate the file using a Shell script.
- In the same shell script add code that **determines if a given column within the csv file contains a value that matches a given regular expression (the regular expression will be input).**
- Check Linux regular expression list reference here.
- The shell script takes as input from the user : **(i) The name of the file, (ii) The column in which to search , (iii) The regular expression to be matched**. The output is a "YES / NO" depending on whether there is matching data present in the file or not.
- Example input :
    - Name of file : "Data.csv"
    - Column number : 4
    - Regular expression : " 5$ " [Find all numbers ending with 5]
- Name your shell script “Assgn1_1g_< groupno>_<roll no. 1>_< roll no. 2>.sh”

Moodle Submission Guidelines:

- Create a single zip file with all the shell scripts (with specified names). Name the zip file : “Assgn1_< groupno>_<roll no. 1>_< roll no. 2>_submission.zip”