Operating Systems Laboratory (CS39002)

Spring Semester 2022-2023

Assignment 1: Familiarization with shell scripting and shell commands

Assignment given on: January 11, 2023

Assignment deadline: January 22, 2023, 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

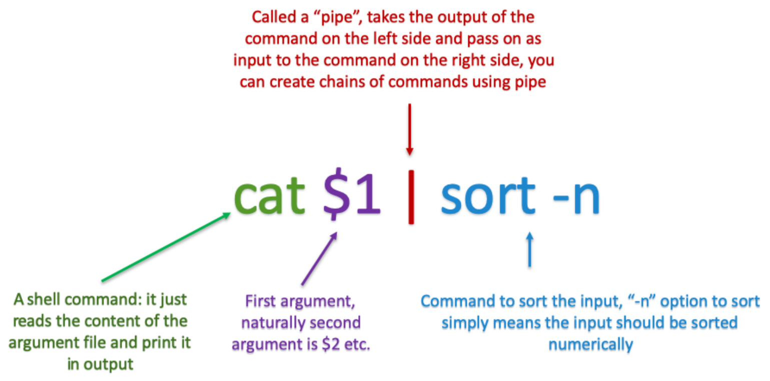
- Open up the terminal.
- Type `cat tmp.txt | sort -n` and press Enter.
- The sorted number will be printed in the terminal

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:


Called a "pipe", takes the output of the command on the left side and pass on as input to the command on the right side, you can create chains of commands using pipe
cat $1 | sort -n
A shell command: it just reads the content of the argument file and print it in output
First argument, naturally second argument is $2 etc.
Command to sort the input, "-n" option to sort simply means the input should be sorted numerically

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
- rev
- 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. White Spaces are not counted.
- 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 - Finding LCM (10 marks)

- You are given a text file (link) containing numerous lines. Each line contains a single number and ends with a newline character.
- Write a shell script which takes the above input, reverses all the numbers and outputs their LCM.
- Name the script as **"Assgn1_1_<groupno>.sh"**
- Example of a text file containing 3 lines:

```
12
36
6
```

The reversed numbers are 21, 63 and 6. The output should be the LCM(21, 63, 6) = 126

- Example Input command:

```
Assgn1_1_XX.sh <input_file>
```

1.2 Pattern Matching - Username Validation (10 marks)

- Write a shell script which takes as input a file (usernames.txt) containing usernames (one username per line) and validates them. It tests the validity of each username and writes (in separate lines) "YES" if the username is valid else "NO" to an output file (say "validation_results.txt")
- A username is valid if the following requirements are satisfied:

    - Number of characters (length) must be between 5 and 20 (both inclusive)
    - Must contain only alphanumeric characters
    - Must contain at least one numeric character
    - Must start with an alphabet
    - Must not contain any of the “invalid words” as substring (case insensitive)
- For your script, create a invalid words file and name it “fruits.txt”. Each line contains a fruit name. the names are from file: fruits.txt
- Some invalid usernames: apPlePie34 (contains invalid word “apple” as a substring, check for invalid words is not case sensitive), 100percent (starts with a number), abc (too short)
- Name the script as **“Assgn1_2_<groupno>.sh”**
- Example Input command:

  Assgn1_2_XX.sh <username_file>

1.3 File Handling / Argument Handling / JSON Parsing (15 marks)

- You are given a directory path (as a command-line argument) which contains several JSONL files, where each line in each file contains a JSON object corresponding to a tweet. Check the definition of JSON object here.
- You have to write a shell script that will convert each JSONL file into a Comma-Separated-Values (CSV) file in a destination folder (given as the 2nd command line argument). The file names will be the same but will end in a “.csv” instead.
- The script should also take as command-line arguments a list of attribute names which are to be extracted (*eg. id_str, created_at, lang*).
- For each JSON object you only need to extract these given attributes and put them into corresponding CSV files as different columns. Thus, each row in a CSV file corresponds to a JSON object in the corresponding JSONL file and each column corresponds to a particular attribute.
- Note that CSV format also has some additional formats for when there are commas present inside a particular text field. See the format rules in the references.
  Name the script as **“Assgn1_3_<groupno>.sh”**
- Example Input command:
  Assgn1_3_XX.sh json_dumps/ csv_dumps/ id_str text retweet_count
- *Example Inputs (with JSONL files) and Outputs are available at this link.*
- References for this assignment:
    - ➢ CMD Line Argument Handling
    - ➢ Parsing JSON in UNIX
    - ➢ CSV File format basic rules
    - ➢ Tweet Object JSON format

1.4. File Manipulation / Argument Handling (10 marks)

- You are given a text file containing numerous lines. Each line includes words, numbers, and special characters and ends with a newline character. You are also given an input word.
- Write a shell script that takes the above input and does the following. If a line contains the input word, then output the line in alternating case, else it remains the same. Do not use loops while converting a line to alternating case. You may use loops elsewhere.

- Name the script as **"Assgn1_4_<groupno>.sh"**
- Sample text file : input4.txt (Take input word as 'kharagpur')
- Alternating Case Example :

  "Just som3 rand0m text with spec!4L CHARS?" is converted to "JuSt SoM3 rAnD0m TeXt WiTh SpEc!4L cHaRs?"

- Example Input:

  Assgn1_4_XX.sh <input_file> <input_word>

1. 5. Pattern matching (10 marks)

- You are given a directory with numerous subdirectories (which could have more subdirectories within them) and files. (Download the data and unzip it).
- Write a shell script which will take the input to the folder and iterate through all the python files in the directory and subdirectories and print the file name along with the path. Also print all the comments in the file along with their line numbers. For multi line comments print the starting line number.
- Name the script as **"Assgn1_5_<groupno>.sh"**
- **Example Input:**

  Assgn1_5_XX.sh <path_to_input_folder>

1.6. File Handling/Sieve of Eratosthenes/Loops/Arrays (10 marks)

- You are given an input file named "input.txt" (link) containing multiple lines with each line containing an integer "n" which lies in the range [1, 10^6].
- Write a shell script that reads n in each line, finds out all the prime divisors of "n" using the Sieve of Eratosthenes algorithm (Ref.), and prints them space separately in a different line in an output file named "output.txt".
- You should use the Sieve of Eratosthenes algorithm only once and consider "n" given in each line as an online query to be processed as you read the input file "input.txt".
- Name the script as **"Assgn1_6_<groupno>.sh"**
- Sample "input.txt":

  5

  22

- Expected "output.txt":

  2 3 5

  2 3 5 7 11 13 17 19

1.7. File Handling / Sorting / Understanding strings (10 marks)

- You are given an input directory path (as a command-line argument) which contains several .txt files, where every file contains multiple names (one per line).
- You are also given an output directory path (it may or may not exist — if it doesn't, create it).

- You have to gather all names (across all files) beginning with the same letter in a single file. Thus, there will be 26 files (created by you inside the output directory) — one for each alphabet. Files can be empty if there is no name beginning with the respective character.
- You also have to ensure that the names inside each of these 26 output files are sorted in lexicographic order (ascending).

  Name the script as **"Assgn1_7_<groupno>.sh"**
- Example Input:
  `Assgn1_7_XX.sh <path_to_input_dir> <path_to_output_dir>`
- References for this assignment:
  - ➢ Lexicographic ordering

1.8. CSV File Manipulation (20 marks)

On the onset of the new year, you and your friends went for a tour outside Kharagpur for a few days. During the tour you and your friends spend money on various things like food, drinks, games, hotel and travel. Since your group is a tech geek, your friends have asked you to create a shell script to track expenses. The shell script essentially manipulates a csv file and should have the following functionalities in order.

- Your shell script should be names as **Assgn1_8_<groupno>.sh**
- Your shell script should create main.csv if that is not present in the script directory.
- The csv should store the following columns

  Date (dd-mm-yy) | Category | Amount | Name

- By default, the script should accept 4 arguments representing a new record (row) in csv and should add the record in csv.
  - For example: **sh Assgn1_8_<groupno>.sh 01-01-23 drinks 250 Mohan**
  - Output: Inserted **01-01-23,drinks,250,Mohan** in main.csv
- Additionally, of course your friends want to know how much they should pay for the trip and other financial information. Your shell script should help them. Specifically, the shell script should accept the following flags :
  - -c 'Category': accepts a category and prints the amount of money spend in that category
  - -n 'name' : accepts a name and print the amount spent by that person
  - -s 'column': sort the csv by column name
  - -h : show help prompt , this must show the name of the utility (you make give any name), Usage or synopsis, and Description. Please refer to the output of 'man mkdir' for reference.

  Please note all these flags are optional, The user may use any number of flags depending upon the need.

- By default all the rows in main.csv should be sorted in chronological order i.e. by date.
  - usage: sh main.sh -c category -n 'name' -s column record [...]
  - Example: sh Assgn1_8_<groupno>.sh -c food -n Mohan -s amount 01-01-23 food 550 Sudarshan

Your script should (in this order)

inserts 01-01-23,food,550,Sudarshan into the csv

Prints the total amount spent on food

Prints the total amount spent by Mohan

Sort the csv according to 'amount' column

NOTE: Insertion always happen first then results are calculated

1. 9 Data Analysis (5 marks)

- Write a script **"Assgn1_9_<groupno>.sh"**
- The script takes a text file as input. The txt file contains the name and the major of the student in each line and the shell script should print the following:
  a. Major names and their count separated by a space, sorted by count descending the number of students. If the count is the same, sort in ascending alphabetical order of the major name.
  b. Names of students having 2 or more majors
  c. Number of students having a single major
- Sample text file: majors.txt with following contents:

```
Arya CS
Babul CS
Chandrani CIVIL
Dinesh MECH
Eeshani ELEC
Arya ELEC
```

Usage: sh Assgn1_9_<groupno>.sh majors.txt

Sample Output:

```
ELEC 2
CIVIL 1
CS 1
MECH 1

Arya
4
```

Explanation: Sorted output of the first task is printed then followed by a space where the students having multiple majors are posted(Here only a single student Arya). Finally 4, the number of students having a single major is printed

- Name the script as **“Assgn1_9_<groupno>.sh”**
- **Example Input:**

```
Assgn1_9_<groupno>.sh <path_to_input_file>
```

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”