

CSc 3320: Systems Programming

Spring 2021

Homework

1: 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: Rafid H. Shaon

Campus ID: rshaon1

Panther #: 002-49-7367

PART 1

Answer the following questions briefly. Provide clear and succinct reasoning.

Points per question = 5

1. Tell the differences between Unix and Linux. Then please list some operating systems (at least three) which belong to Unix but not Linux.

UNIX, powerful operating system, not owned by any single company, generally expensive and used primarily by large companies who have legacy programs or who will pay more for more guaranteed uptime. Runs on many types of processors. Linux, a "clone" of UNIX that is free to use, has the power of UNIX and is licensed using the GNU license.

- Linux is free to use, while Unix is a licensed OS.
- Linux supported file systems are Ext2, Ext3, Ext4, Jfs, ReiserFS, Xfs, Btrfs, FAT, FAT32, NTFS. Unix's are fs, gpfs, hfs, hfs+, ufs, xfs, zfs. As you can see Unix has fewer supported file systems.
- Bash (Bourne Again Shell) is the default shell for Linux. Bourne Shell is the default shell for Unix.
- Linux uses KDE and Gnome. Other GUI supported are LXDE, Xfce, Unity, Mate. Unix was initially a command based OS. Most of the unix distributions now have Gnome.

2. What is the pipe mechanism in UNIX? And show one command using pipe and explain how the pipe works in it?

Pipe is used to combine two or more commands, and in this, the output of one command acts as input to another command, and this command's output may act as input to the next command and so on.

Example: ----->

ls | wc -l

The above command is connecting two individual commands which are ls and wc using a pipe(|).

The **ls** command is used to show the files, directories and links present in the current directory and **wc** is used to count these file system objects.

3. In a Linux system, you can issue the command **ls /** to check the sub directories under root. Please describe the meanings of directory **/bin**, **/dev**, **/boot**, **/usr**, **/etc**, **/mnt**, **/sbin**, **/var** separately. For example, you can say that **/bin** contains binary executable files.

/bin - User Binaries

The above command contains the binary executable files. The single-user mode commands are listed under these type of commands.

The examples of the commands which can be listed under the command **/bin** can be listed as follows:

- ps
- ls
- grep

/dev - Device Files

- The device files are listed under this command (**/dev**).
- The terminal devices such as usb or any device connected to the system can be included under this command.
- The few examples of the commands which can be listed under this command are **/ttyl**, **/dev/usbmon0**

/boot - Boot Loader Files

This command includes boot loader files. For example: **initrd.img-2.6.32-24-generic** etc.

/usr - User Programs

This command contains libraries, binaries and source code for second level program. The various commands which can be used with this command are as follows:

- **/usr/bin**: binary files for user programs.
- **/usr/sbin**: includes binary executable files for system administrators rather than users.
- **/usr/lib**: includes libraries for commands like **/usr/bin** and **/usr/sbin**.
- **/usr/local**: includes programs installed from source such as apache is installed from source and will be stored as **/usr/local/apache2**.

/etc - Configuration Files

This command contains script for start and shutdown of individual programs. Configuration files are the major part of this command. For example: /etc/logrotate.conf.

/mnt - Mount Directory

This command can be used for mount directory where sys admins can mount file systems.

/sbin - System Binaries

This command is similar to /bin. It also contains binary executables but it is for system administrators not for users. For example: reboot, fdisk etc.

/var - Variable Files

This command includes files whose content can be grown or which are variable in nature. Example: emails (/var/mail), print files(/var/spool) etc.

4. What is the meaning of Multitask and Multi-user in a Unix system?

Unix can do many jobs at once, dividing the processor's time between the tasks so quickly that it looks as if everything is running at the same time. This is called multitasking.

Unix is a Multi-user operating system is a computer operating system which allows multiple users to access the single system with one operating system on it. It is generally used on large mainframe computers.

5. What does -rwxr-xr-x mean in terms of permissions for a file? What is the exact unix command (with the octal representation) for changing the permissions to this setting?

- -rwxr-xr-x (755) -- The user has read, write and execute permissions; the group and others can only read and execute.
- Here you can see that the directory has permission values of 755 (read + write + execute or 4 + 2 + 1, read + execute or 4 + 1, and read + execute or 4 + 1).
- You can change permissions with the **chmod** command by using letters or numbers. Type **chmod *permissions* file** to change permissions of a file or directory.

6. In class, you have learned the meaning of read, write and execute permission for regular files. However, these permissions are also applied to directories. So please describe the meaning of read, write, and execute permission for directory.

Read permission means that the user may see the contents of a directory (e.g. use `ls` for this directory.) Write permission means that a user may create files in the directory. Execute permission means that the user may enter the directory (i.e. make it his current directory.)

Part II-a

Regular Expression

Find outcomes for each given basic/extended regular expression (maybe multiple correct answers) and describe the pattern of matched string for 3), 4), 5), 6), 11).

Points per question: 2.5

Example:

'ab+a' (extended regex)

a) ababa b) aba c) abba d) aabbaa e) aa

Answer: b,c ; ***Pattern :*** The matched string should begin and end with 'a' and 'b' occurs at least once between leading and ending 'a')

Note: 7) to 10) are basic regexes; Note: 11) to 18) are extended regexes.

7) 'a[ab]*a'

The above regular expression generates a string that contains the symbol 'a' at the start and end of the string. In UNIX, [] matches any one of the enclosed characters and '*' matches the 0 or more occurrences of the preceding character

Answer: ababa, aaba, aabbaa, aa

8) 'a(bc)?'

The above regular expression generates a string that starts with 'a' and ends with 0 or 1 occurrence of the substring 'bc'. In UNIX, () allows to group several characters as one (i.e., sub pattern) and '?' matches the 0 and 1 occurrences of the preceding character.

Answer: abc, a

9) '[ind]*'

The above regular expression generates a string that starts with any character and ends with 0 or more occurrences of the any of the enclosed characters. In UNIX, [] matches any one of the enclosed characters and '*' matches the 0 or more occurrences of the preceding character. The '.' matches any single character.

Answer: wind, end

10) '[a-z]+[a-z]'

The above regular expression generates a string that starts with 1 or more occurrences of the any one of the enclosed characters and ends with any one of the enclosed characters. In UNIX, [] matches any one of the enclosed characters and '+' matches the 1 or more occurrences of the preceding character.

Answer: xx, azaz

11) '[a-z](\[a-z])+'

The above regular expression generates a string that starts with any one of the enclosed characters and ends with 1 or more occurrences of the sub pattern. Here, the sub pattern starts with '+' and ends with any one of the enclosed characters. In UNIX, [] matches any one of the enclosed characters and '+' matches the 1 or more occurrences of the preceding character. The '\' turns off the special meaning of the next character.

Answer: a+b+c, x+a

12) 'a.[bc]+'

The above regular expression generates a string that starts with 'a' then any single character and then 1 or more occurrences of the enclosed characters. In UNIX, [] matches any one of the enclosed characters and '+' matches the 1 or more occurrences of the preceding character. The '.' matches any single character.

Answer: azbc, azbcbc, acc

13) 'a.[0-9]'

The above regular expression generates a string that starts with 'a' then any single character and then any one of the enclosed characters within the range. In UNIX, [] matches any one of the enclosed characters and '.' matches any single character. The '-' is used to define the range.

Answer: a01

14) '[a-z]+[\\.\?!]'

The above regular expression generates a string that starts with one or more occurrences of any one of the enclosed characters within the range and ends with any one of the enclosed characters. In UNIX, [] matches any one of the enclosed characters and '.' matches any single character. The '\\' turns off the special meaning of the next character.

Answer: good!, hard?

15) '[a-z]+[\\.\?!]\\s*[A-Z]'

The above regular expression generates a string that starts with one or more occurrences of any one of the enclosed characters within the range then any number of spaces and then ends with any one of the enclosed characters within the range. In UNIX, [] matches any one of the enclosed characters and '.' matches any single character. The '\\' turns off the special meaning of the next character and '*' matches the 0 or more occurrences of the preceding character.

Answer: book. Z

16) '(very)+(cool)?(good|bad) weather'

The above regular expression generates a string that starts with 'very' then 0 or 1 occurrences of 'cool' then 'good' or 'bad' and then ends with 'weather'.

Answer: very good weather, very cool bad weather

17) '-?[0-9]+'

The above regular expression generates a string that starts with 0 or 1 occurrences of the '-' then ends with 1 or more occurrences of the enclosed characters within the range.

Answer: 3312, -2231

18) '-?[0-9]*\\.?[0-9]*'

The above regular expression generates a string that starts with 0 or 1 occurrences of the '-' then 0 or more occurrences of the enclosed characters within the range then 0 or 1 occurrences of the '.' then ends with 0 or more occurrences of the enclosed characters within the range.

Answer: 3312, -2231, 0.5

Part II-b

Regular Expression

Write down the extended regular expression for following questions. E.g. Social security number in the format of 999-99-9999. Answer:

`[0-9]{3}-[0-9]{2}-[0-9]{4}`

Points per question: 5

19) Valid URL beginning with "http://" and ending with ".edu" (e.g. <http://cs.gsu.edu>, <http://gsu.edu>)

`(http:\\\\)[a-z]*\\.?[a-z]*(.edu)`

20) Non-negative integers. (e.g. 0, +1, 3320)

`\\+?[0-9]+`

21) A valid absolute pathname in Unix (e.g. /home/ylong4, /test/try.c)

`\\[a-z]+\\[a-z]+[0-9]*\\.?[a-z]*`

22) Identifiers which can be between 1 and 10 characters long, must start with a letter or an underscore. The following characters can be letters or underscores or digits. (e.g. number, _name1, isOK).

`([_]*[a-z]*[_]*[a-z]*[0-9]*[A-Z]*){1,10}`

23) Phone number in any of the following formats: 9999999999,999-999-9999, (999)-999-9999. (Note: all of these formats should be matched by a single regular expression)

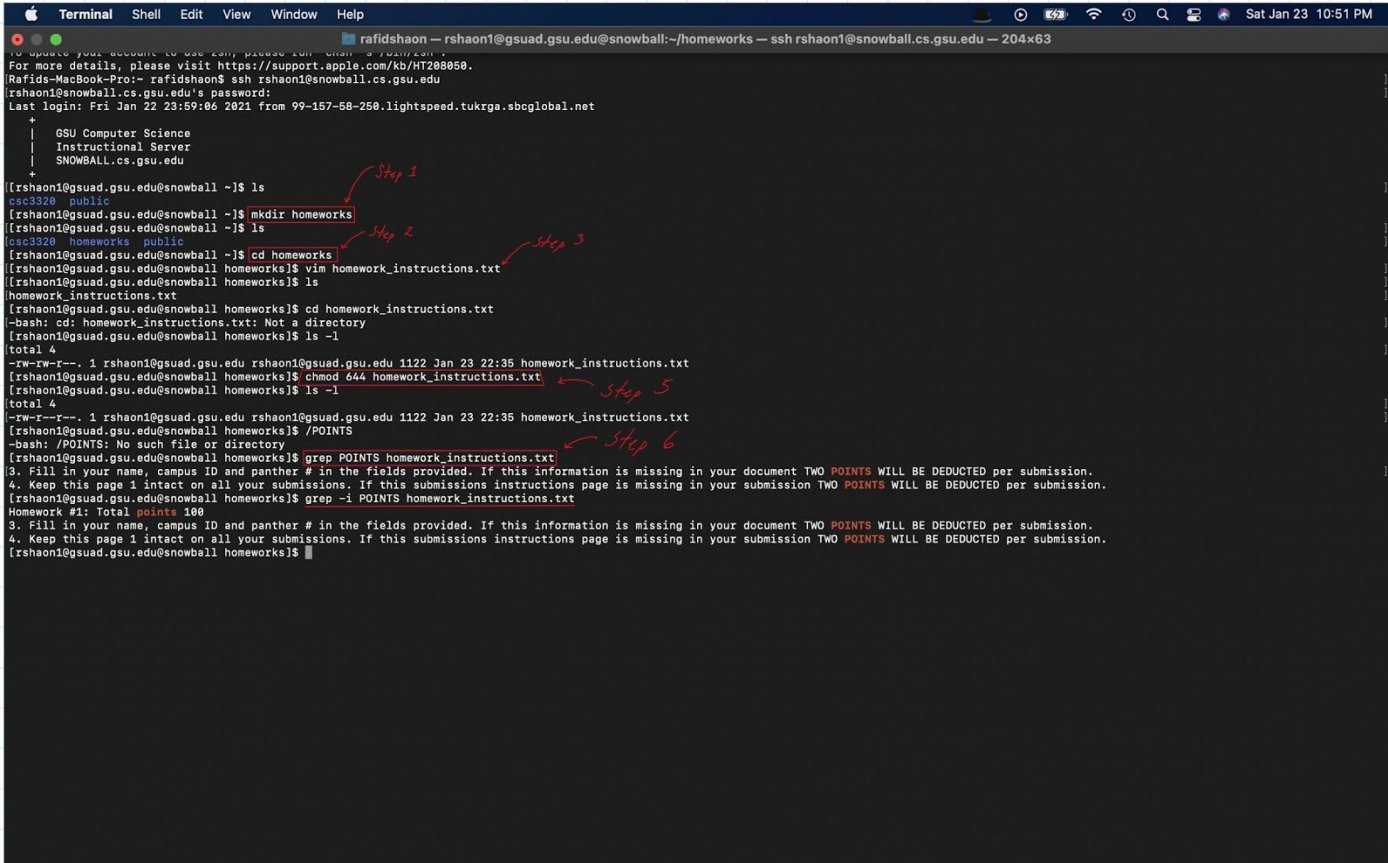
`[([0-9]*{1,10})([0-9]*{1,3}-[0-9]*{1,3}-[0-9]*{1,4})|(\\([0-9]*{1,3}\\)-[0-9]*{1,3}-[0-9]*{1,4})]`

Part III

Programming

Points per question: 15

24. Create a file named `homework_instructions.txt` using VI editor and type in it all the submission instructions from page1 of this document. Save the file in a directory named *homeworks* that you would have created. Set the permissions for this file such that only you can edit the file while anybody can only read. Find and list (on the command prompt) all the statements that contain the word POINTS. Submit your answer as a description of what you did in a sequential manner (e.g. Step1 ... Step 2... and so on..). Add a screenshot to your answer as a proof of evidence.



```
Terminal Shell Edit View Window Help
rafidshaon — rshaon1@gsuad.gsu.edu@snowball:~/homeworks — ssh rshaon1@snowball.cs.gsu.edu — 204x63

To update your account to use 256-bit please run 'ssh-keygen -s /usr/ssh/rsa1024'
For more details, please visit https://support.apple.com/kb/HT208050.
Rafids-MacBook-Pro:~ rafidshaon$ ssh rshaon1@snowball.cs.gsu.edu
rshaon1@snowball.cs.gsu.edu's password:
Last login: Fri Jan 22 23:59:06 2021 from 99-157-58-250.lightspeed.tukrga.sbcglobal.net

+
| GSU Computer Science
| Instructional Server
| SNOWBALL.cs.gsu.edu
+

[rshaon1@gsuad.gsu.edu@snowball ~]$ ls
csc3320 public
[rshaon1@gsuad.gsu.edu@snowball ~]$ mkdir homeworks
[rshaon1@gsuad.gsu.edu@snowball ~]$ ls
csc3320 homeworks public
[rshaon1@gsuad.gsu.edu@snowball ~]$ cd homeworks
[rshaon1@gsuad.gsu.edu@snowball homeworks]$ vim homework_instructions.txt
[rshaon1@gsuad.gsu.edu@snowball homeworks]$ ls
homework_instructions.txt
[rshaon1@gsuad.gsu.edu@snowball homeworks]$ cd homework_instructions.txt
-bash: cd: homework_instructions.txt: Not a directory
[rshaon1@gsuad.gsu.edu@snowball homeworks]$ ls -l
total 4
-rw-rw-r--r. 1 rshaon1@gsuad.gsu.edu rshaon1@gsuad.gsu.edu 1122 Jan 23 22:35 homework_instructions.txt
[rshaon1@gsuad.gsu.edu@snowball homeworks]$ chmod 644 homework_instructions.txt
[rshaon1@gsuad.gsu.edu@snowball homeworks]$ ls -l
total 4
-rw-r--r--r. 1 rshaon1@gsuad.gsu.edu rshaon1@gsuad.gsu.edu 1122 Jan 23 22:35 homework_instructions.txt
[rshaon1@gsuad.gsu.edu@snowball homeworks]$ /POINTS
-bash: /POINTS: No such file or directory
[rshaon1@gsuad.gsu.edu@snowball homeworks]$ grep POINTS homework_instructions.txt
(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.
[rshaon1@gsuad.gsu.edu@snowball homeworks]$ grep -i POINTS homework_instructions.txt
Homework #1: Total points 100
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.
[rshaon1@gsuad.gsu.edu@snowball homeworks]$
```

```
ESC 3320: Systems Programming
Spring 2021
Homework #1: 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:

Campus ID:

Panther #:

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

~

"homework_instructions.txt" 23L, 1122C

1,1

All

Step 4

Step 1: I made a directory named homeworks. Right after I typed “ls” to see if it was actually there.

Step 2: So I could go into the directory

Step 3: Now that I’m in homeworks I’m using VI editor to create the .txt

Step 4: I got into the editor and I was in “Normal Mode” I pressed “i” and now I’m in “Insert Mode” I went to the first page of the HW copied the instructions and everything. Then I pasted it into the .txt . Right after I pressed “Esc” so I could escape back into the “Normal Mode”. Now I typed “:wq” this makes sure to write and quit (save and exit!).

I “ls -l” to see the permission listing of the file.

Step 5: I change the permission so ‘me’ the User can be the only one to read and write on the .txt, while everyone else can just read it. I “ls -l” again to see if the changes I made were actually there.

Step 6: I go to find the word POINTS (case sensitive) in homeworks_instructions.txt and see all the statements with that word in it.

[The Red Underline: It's just me doing the same thing in Step 6 for the most part with the one exception where I put "-i" so I can make it case insensitive and see the word POINTS in general with their statements.]