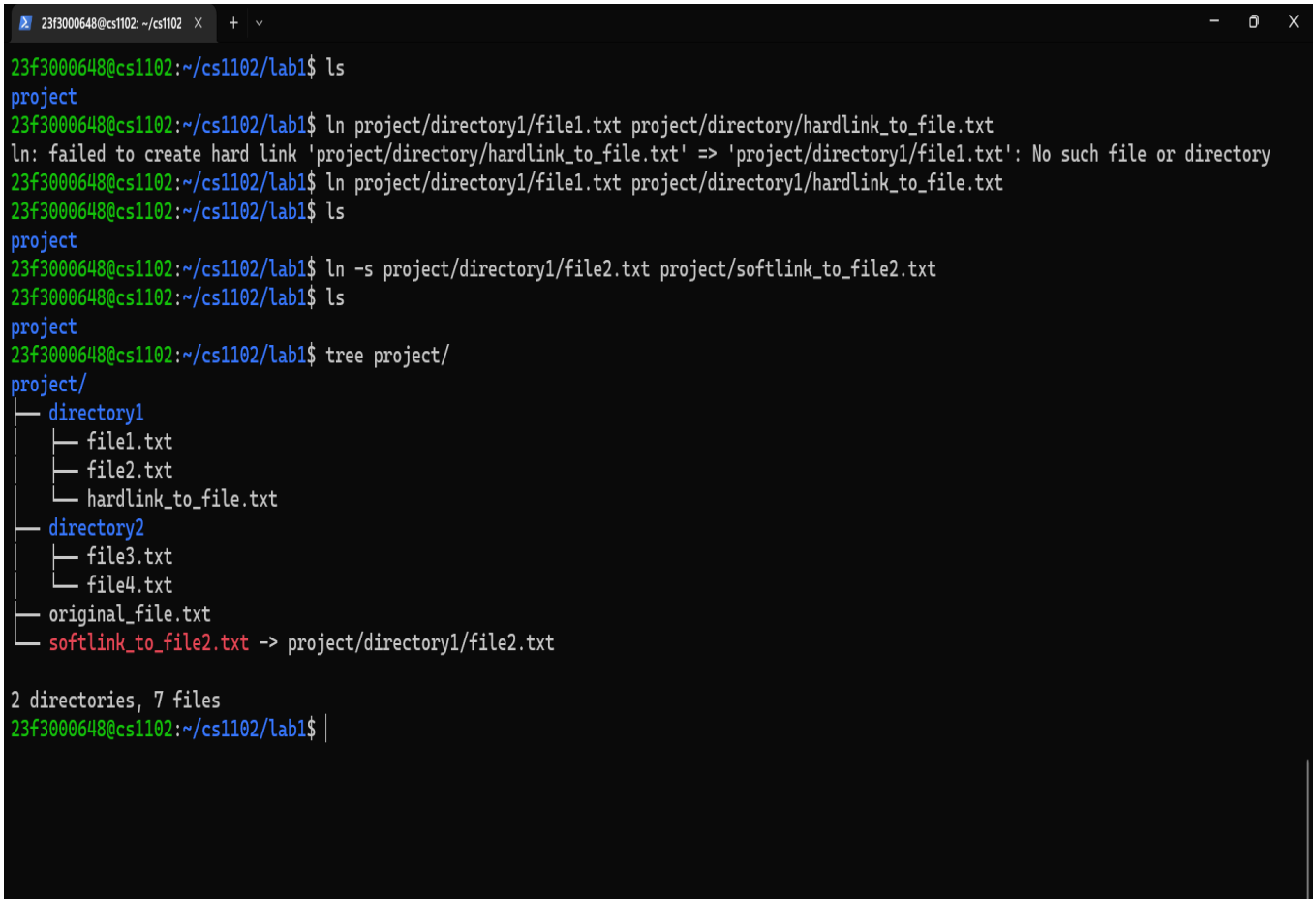
Document Title: Linux Scripting and Command Line Proficiency

Author: Ram Tripathi (23F3000648)

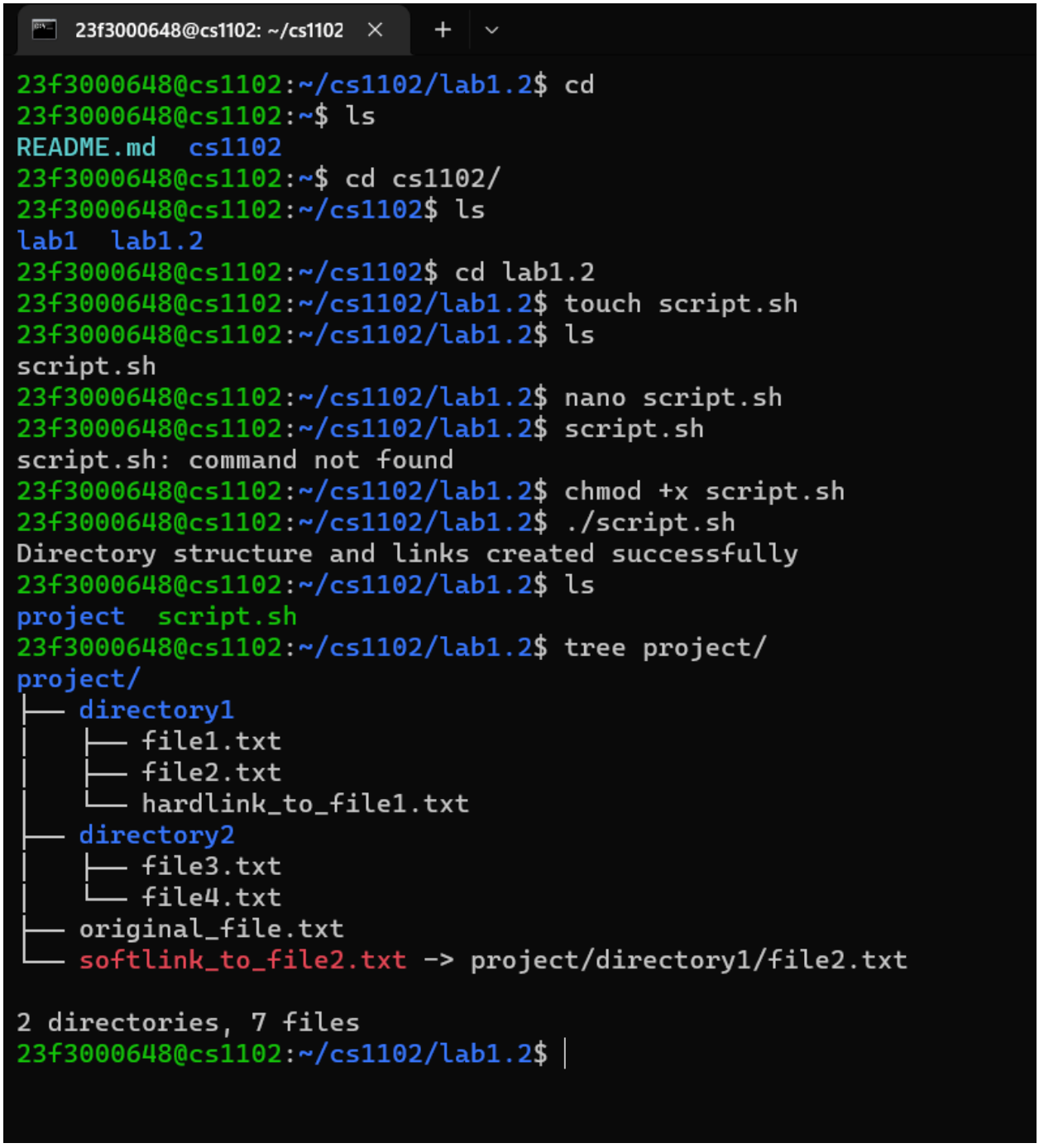
Description:
This document showcases my proficiency in Linux scripting and command-line operations. It includes practical examples of shell scripting, file manipulation, process management, networking commands, and system administration tasks. The content demonstrates my ability to automate tasks, optimize workflows, and efficiently navigate the Linux environment. It has been part of my curriculum for linux course at Indian Institute of Technology, Madras.

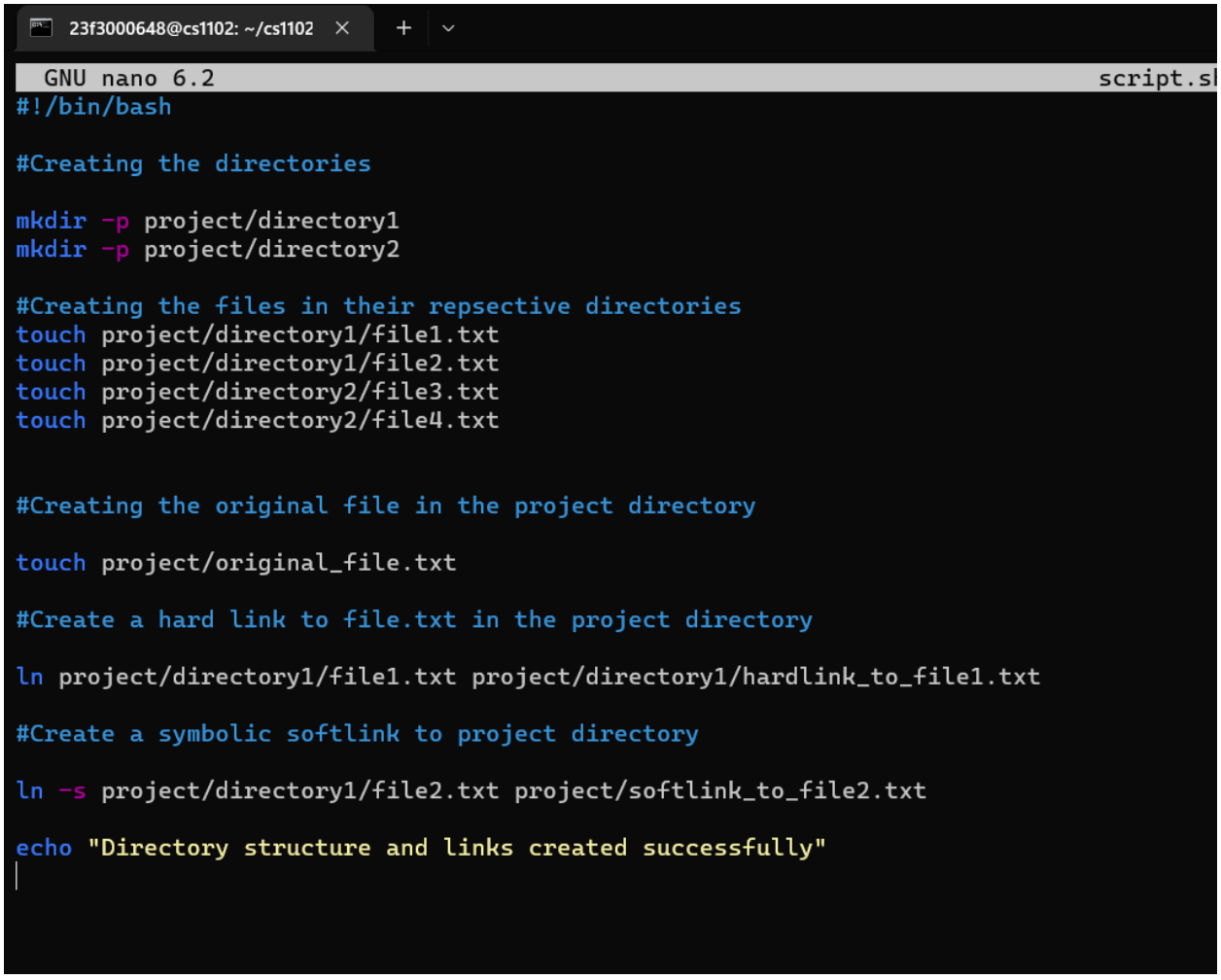
Lab 1 Activities

LAB: 1.1 Creating Directory Structure:

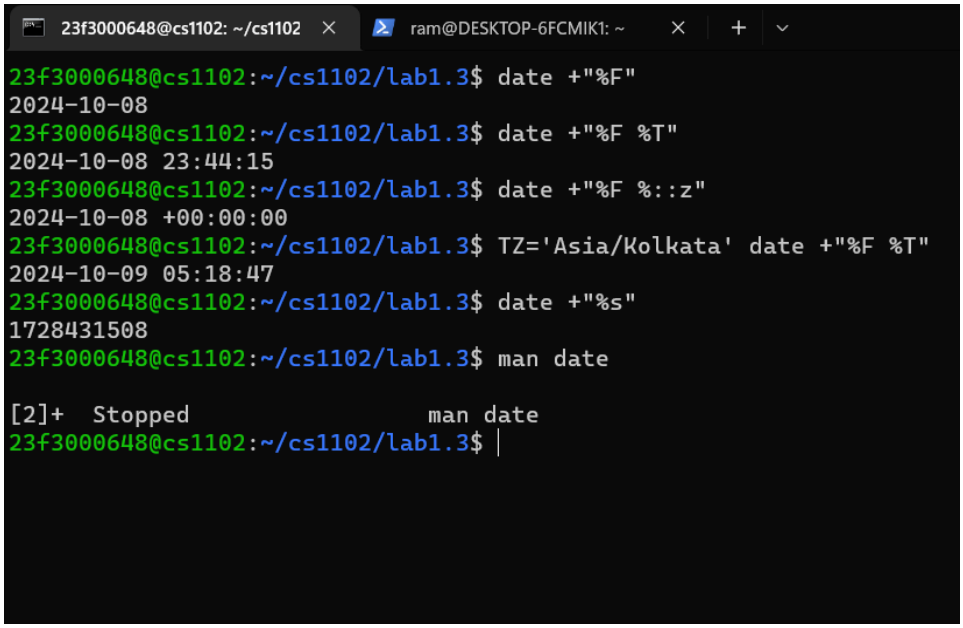


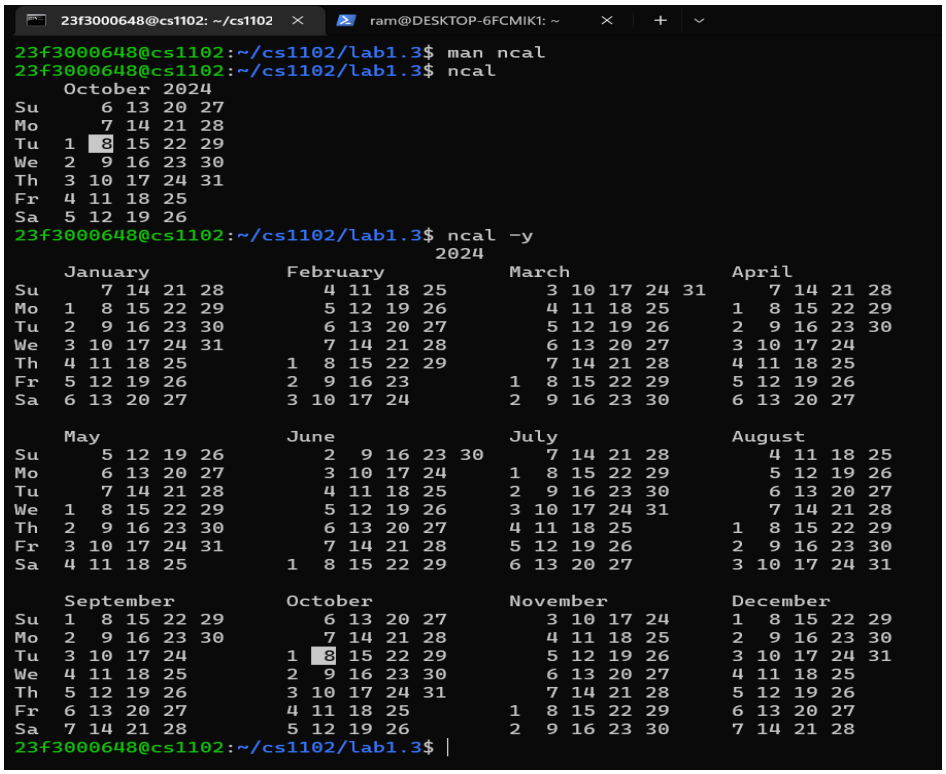
LAB: 1.2 Creating Directory Structure using a script:

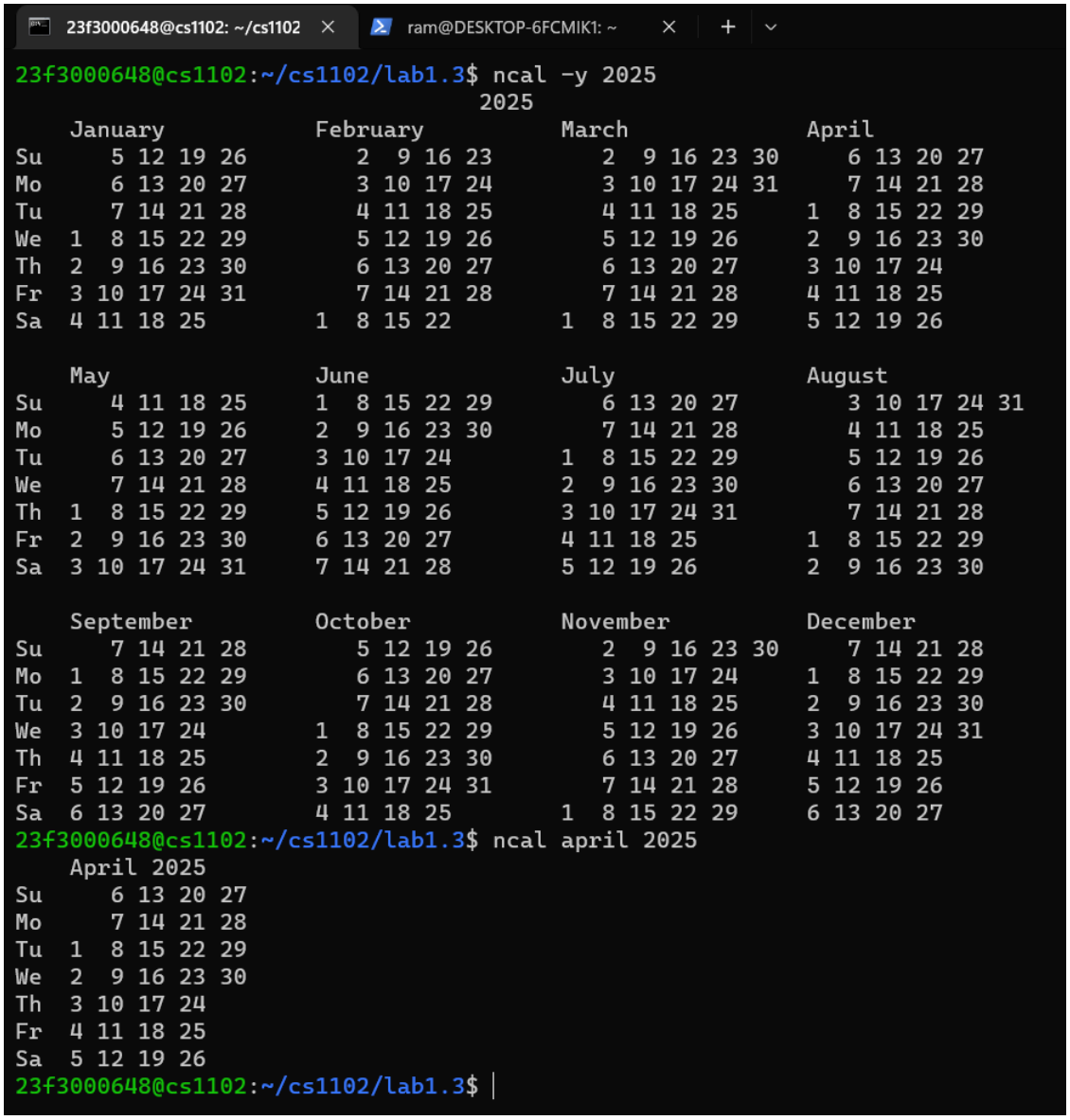

23f3000648@cs1102: ~/cs1102
23f3000648@cs1102:~/cs1102/lab1.2$ cd
23f3000648@cs1102:~$ ls
README.md cs1102
23f3000648@cs1102:~$ cd cs1102/
23f3000648@cs1102:~/cs1102$ ls
lab1 lab1.2
23f3000648@cs1102:~/cs1102$ cd lab1.2
23f3000648@cs1102:~/cs1102/lab1.2$ touch script.sh
23f3000648@cs1102:~/cs1102/lab1.2$ ls
script.sh
23f3000648@cs1102:~/cs1102/lab1.2$ nano script.sh
23f3000648@cs1102:~/cs1102/lab1.2$ script.sh
script.sh: command not found
23f3000648@cs1102:~/cs1102/lab1.2$ chmod +x script.sh
23f3000648@cs1102:~/cs1102/lab1.2$ ./script.sh
Directory structure and links created successfully
23f3000648@cs1102:~/cs1102/lab1.2$ ls
project script.sh
23f3000648@cs1102:~/cs1102/lab1.2$ tree project/
project/
├── directory1
│   ├── file1.txt
│   ├── file2.txt
│   └── hardlink_to_file1.txt
├── directory2
│   ├── file3.txt
│   └── file4.txt
├── original_file.txt
└── softlink_to_file2.txt -> project/directory1/file2.txt
2 directories, 7 files
23f3000648@cs1102:~/cs1102/lab1.2$



LAB: 1.3 Date and Ncal Commands


23f3000648@cs1102: ~/cs1102
ram@DESKTOP-6FCMIK1: ~
23f3000648@cs1102:~/cs1102/lab1.3$ date +"%F"
2024-10-08
23f3000648@cs1102:~/cs1102/lab1.3$ date +"%F %T"
2024-10-08 23:44:15
23f3000648@cs1102:~/cs1102/lab1.3$ date +"%F %::z"
2024-10-08 +00:00:00
23f3000648@cs1102:~/cs1102/lab1.3$ TZ='Asia/Kolkata' date +"%F %T"
2024-10-09 05:18:47
23f3000648@cs1102:~/cs1102/lab1.3$ date +"%s"
1728431508
23f3000648@cs1102:~/cs1102/lab1.3$ man date
[2]+ Stopped man date
23f3000648@cs1102:~/cs1102/lab1.3$


23f3000648@cs1102: ~/cs1102
ram@DESKTOP-6FCMIK1: ~
23f3000648@cs1102:~/cs1102/lab1.3$ man ncal
23f3000648@cs1102:~/cs1102/lab1.3$ ncal
October 2024
Su 6 13 20 27
Mo 7 14 21 28
Tu 1 8 15 22 29
We 2 9 16 23 30
Th 3 10 17 24 31
Fr 4 11 18 25
Sa 5 12 19 26
23f3000648@cs1102:~/cs1102/lab1.3$ ncal -y
2024
January February March April
Su 7 14 21 28 4 11 18 25 3 10 17 24 31 7 14 21 28
Mo 1 8 15 22 29 5 12 19 26 4 11 18 25 1 8 15 22 29
Tu 2 9 16 23 30 6 13 20 27 5 12 19 26 2 9 16 23 30
We 3 10 17 24 31 7 14 21 28 6 13 20 27 3 10 17 24
Th 4 11 18 25 1 8 15 22 29 7 14 21 28 4 11 18 25
Fr 5 12 19 26 2 9 16 23 1 8 15 22 29 5 12 19 26
Sa 6 13 20 27 3 10 17 24 2 9 16 23 30 6 13 20 27
May June July August
Su 5 12 19 26 2 9 16 23 30 7 14 21 28 4 11 18 25
Mo 6 13 20 27 3 10 17 24 1 8 15 22 29 5 12 19 26
Tu 7 14 21 28 4 11 18 25 2 9 16 23 30 6 13 20 27
We 1 8 15 22 29 5 12 19 26 3 10 17 24 31 7 14 21 28
Th 2 9 16 23 30 6 13 20 27 4 11 18 25 1 8 15 22 29
Fr 3 10 17 24 31 7 14 21 28 5 12 19 26 2 9 16 23 30
Sa 4 11 18 25 1 8 15 22 29 6 13 20 27 3 10 17 24 31
September October November December
Su 1 8 15 22 29 6 13 20 27 3 10 17 24 1 8 15 22 29
Mo 2 9 16 23 30 7 14 21 28 4 11 18 25 2 9 16 23 30
Tu 3 10 17 24 1 8 15 22 29 5 12 19 26 3 10 17 24 31
We 4 11 18 25 2 9 16 23 30 6 13 20 27 4 11 18 25
Th 5 12 19 26 3 10 17 24 31 7 14 21 28 5 12 19 26
Fr 6 13 20 27 4 11 18 25 1 8 15 22 29 6 13 20 27
Sa 7 14 21 28 5 12 19 26 2 9 16 23 30 7 14 21 28
23f3000648@cs1102:~/cs1102/lab1.3$



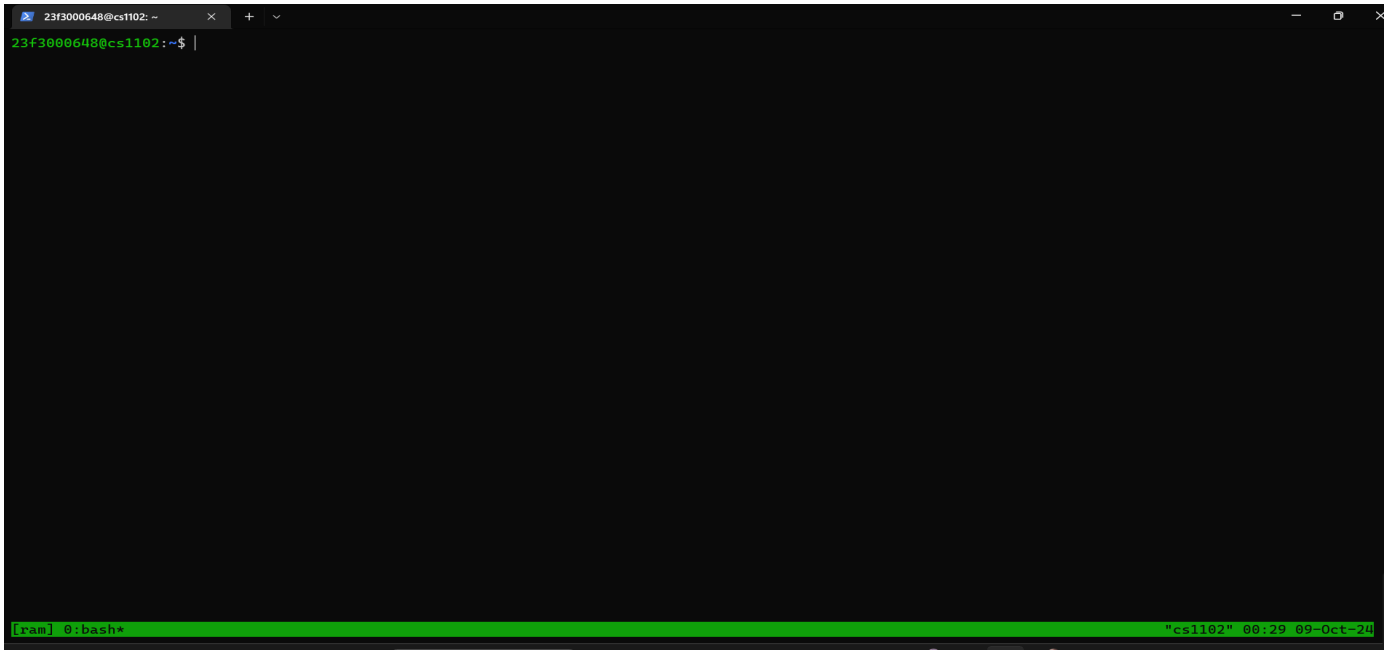
LAB: 1.4 Brace Expansion and Echo Command

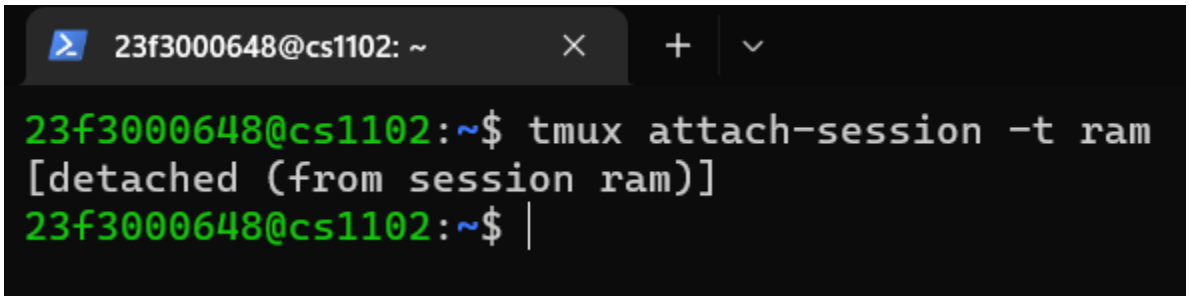
```
23f3000648@cs1102: ~/cs1102   ram@DESKTOP-6FCMIK1: ~
23f3000648@cs1102:~/cs1102/lab1.3$ echo {1..5}
1 2 3 4 5
23f3000648@cs1102:~/cs1102/lab1.3$ echo {a..e}
a b c d e
23f3000648@cs1102:~/cs1102/lab1.3$ echo {1..5}(a..e}
-bash: syntax error near unexpected token `('
23f3000648@cs1102:~/cs1102/lab1.3$ echo {1..5}{a..e}
1a 1b 1c 1d 1e 2a 2b 2c 2d 2e 3a 3b 3c 3d 3e 4a 4b 4c 4d 4e 5a 5b 5c 5d 5e
23f3000648@cs1102:~/cs1102/lab1.3$ echo {5..1}{e..a}
5e 5d 5c 5b 5a 4e 4d 4c 4b 4a 3e 3d 3c 3b 3a 2e 2d 2c 2b 2a 1e 1d 1c 1b 1a
23f3000648@cs1102:~/cs1102/lab1.3$ echo {1..5}{a,c,e}
1a 1c 1e 2a 2c 2e 3a 3c 3e 4a 4c 4e 5a 5c 5e
23f3000648@cs1102:~/cs1102/lab1.3$ echo {z..a}
z y x w v u t s r q p o n m l k j i h g f e d c b a
23f3000648@cs1102:~/cs1102/lab1.3$
```

LAB: 1.5 Tmux

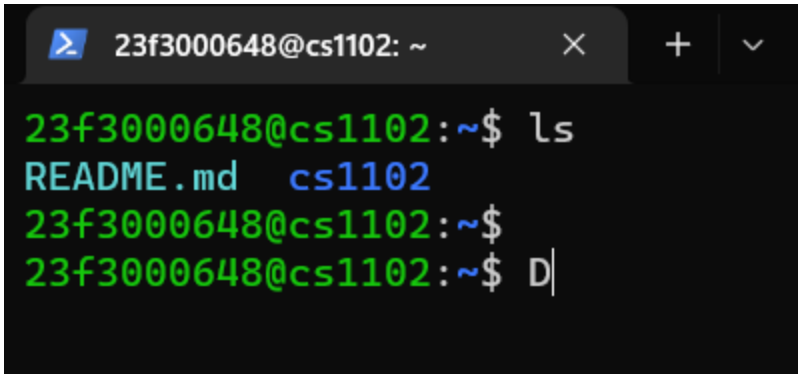
```
23f3000648@cs1102: ~
23f3000648@cs1102:~$ tmux
```

```
23f3000648@cs1102: ~
23f3000648@cs1102:~$ tmux new-session -s shyam
[detached (from session shyam)]
23f3000648@cs1102:~$ tmux list-sessions
0: 1 windows (created Wed Oct  9 00:14:36 2024)
1: 1 windows (created Wed Oct  9 00:20:55 2024)
2: 1 windows (created Wed Oct  9 00:21:56 2024)
ram: 1 windows (created Wed Oct  9 00:23:06 2024)
shyam: 1 windows (created Wed Oct  9 00:25:28 2024)
23f3000648@cs1102:~$
```


23f3000648@cs1102: ~
23f3000648@cs1102:~$
[ram] 0:bash*
"cs1102" 00:29 09-Oct-24

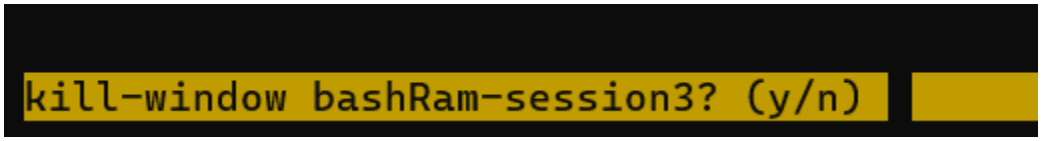

23f3000648@cs1102: ~
23f3000648@cs1102:~$ tmux attach-session -t ram
[detached (from session ram)]
23f3000648@cs1102:~$


[ram] 0:bash 1:bash 2:bash- 3:bash*
"cs1102" 00:32 09-Oct-24

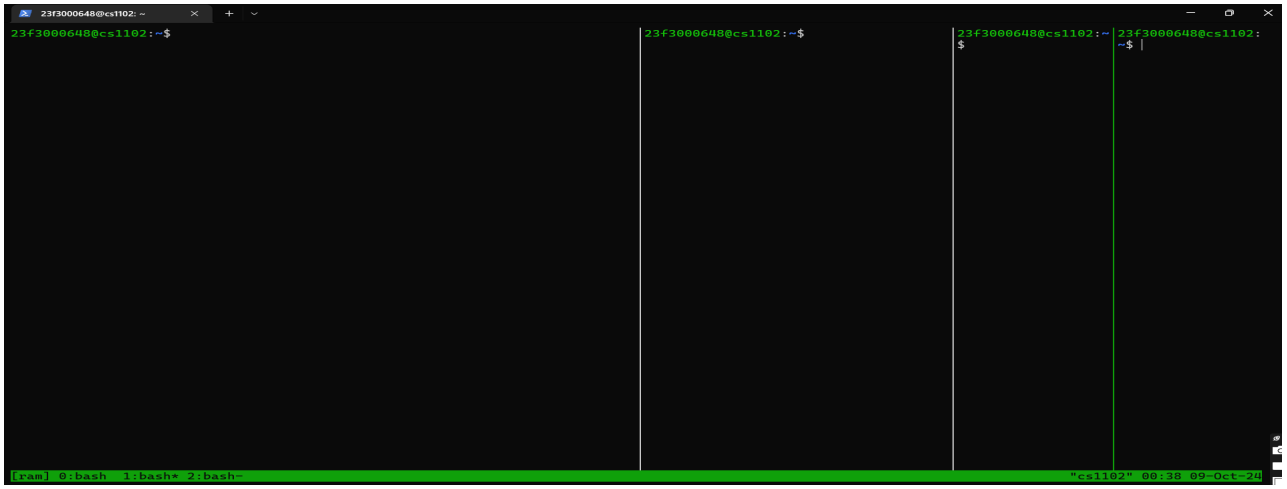

23f3000648@cs1102: ~
23f3000648@cs1102:~$ ls
README.md cs1102
23f3000648@cs1102:~$
23f3000648@cs1102:~$ D

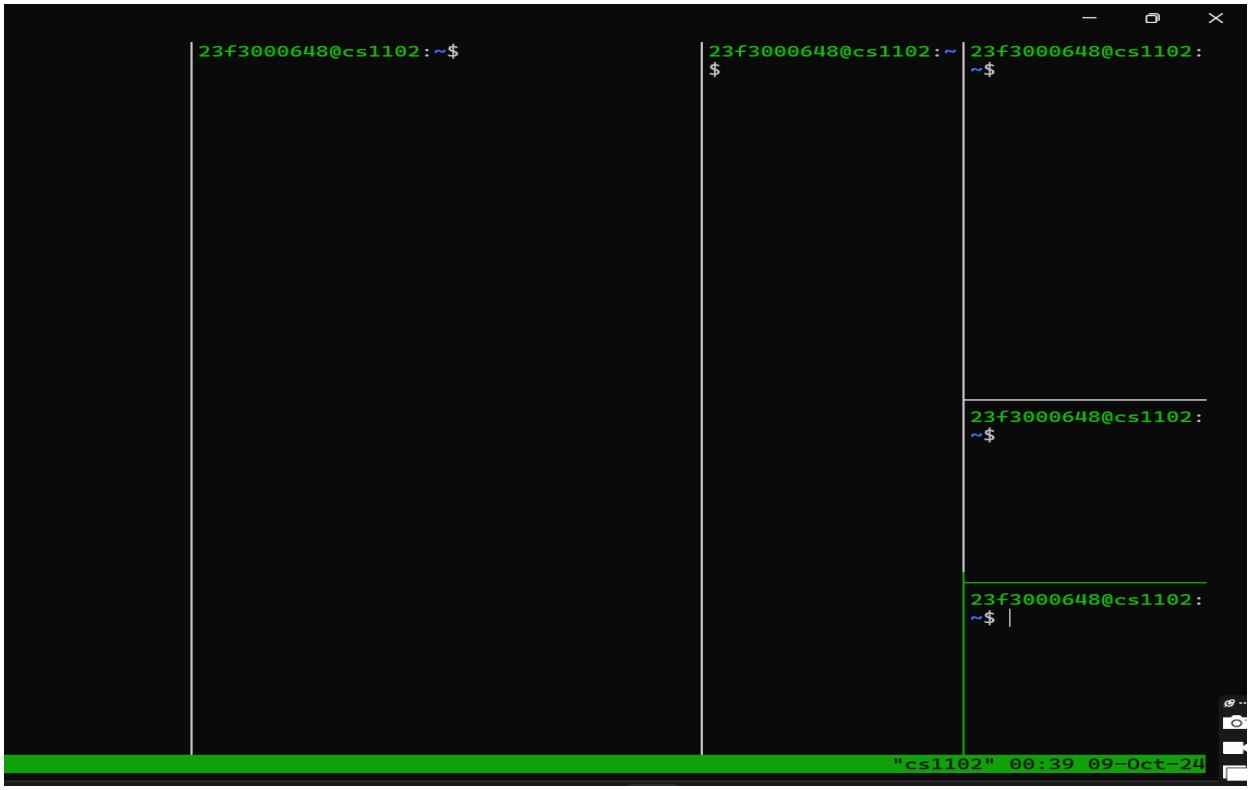

(rename-window) bashRam-session3

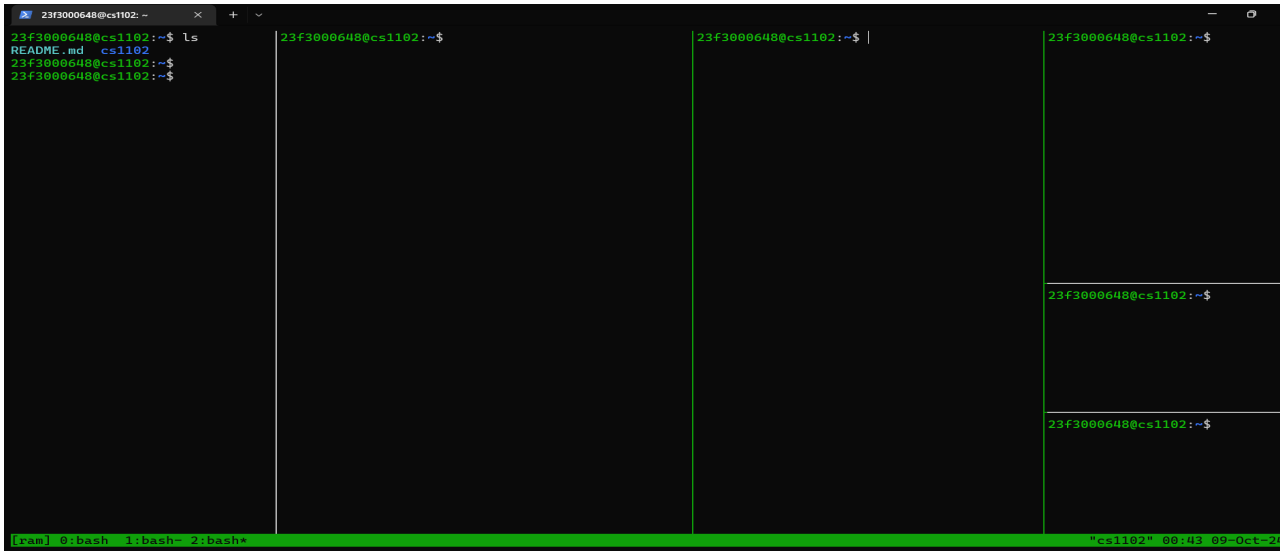

[ram] 0:bash 1:bash 2:bash- 3:bashRam-session3*

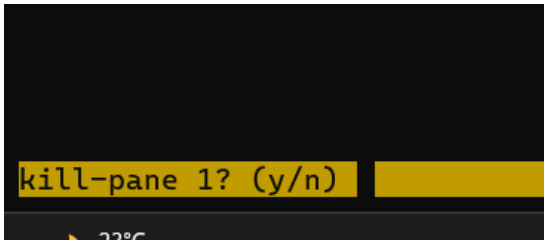

kill-window bashRam-session3? (y/n)


[ram] 0:bash 1:bash- 2:bash*


23f3000648@cs1102:~$
23f3000648@cs1102:~$
23f3000648@cs1102:~$
23f3000648@cs1102:~$
[ram] 0:bash 1:bash* 2:bash-
"cs1102" 00:38 09-Oct-24

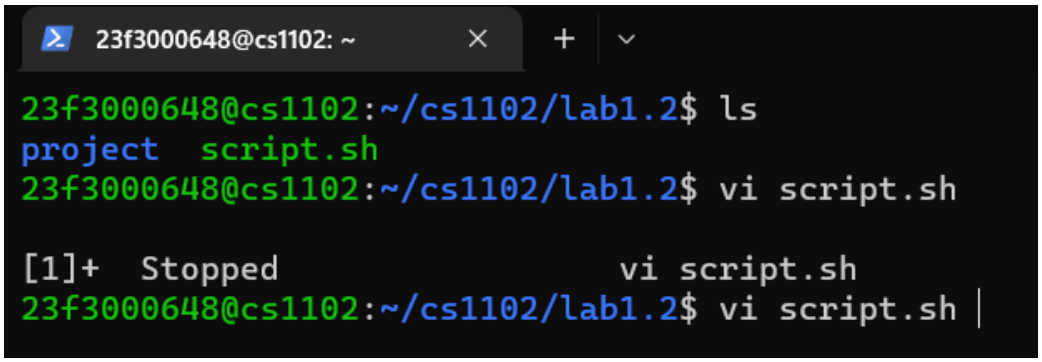

23f3000648@cs1102:~$
23f3000648@cs1102:~$
23f3000648@cs1102:~$
23f3000648@cs1102:~$
23f3000648@cs1102:~$
"cs1102" 00:39 09-Oct-24


23f3000648@cs1102: ~
23f3000648@cs1102:~$ ls
README.md cs1102
23f3000648@cs1102:~$
23f3000648@cs1102:~$
23f3000648@cs1102:~$
23f3000648@cs1102:~$
23f3000648@cs1102:~$
23f3000648@cs1102:~$
23f3000648@cs1102:~$
[ram] 0:bash 1:bash- 2:bash*
"cs1102" 00:43 09-Oct-2

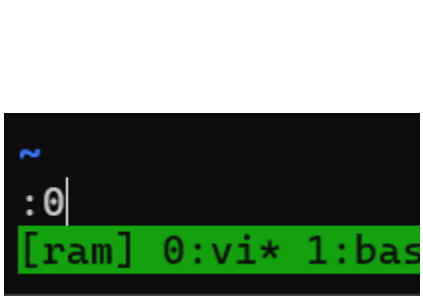


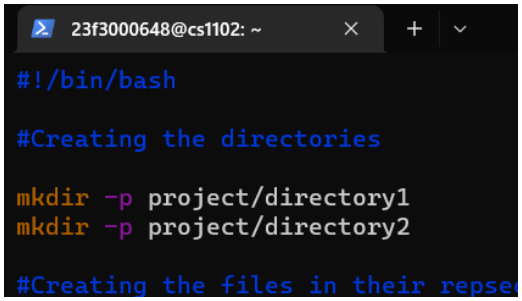
LAB: 1.6 Vi Editor

1.6.1

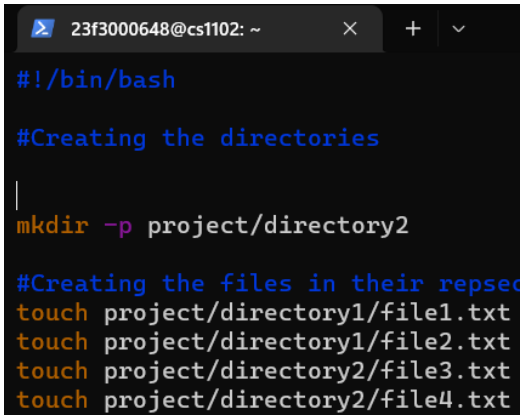


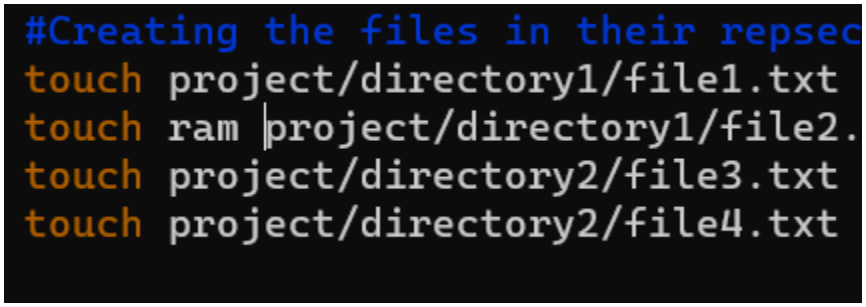
1.6.2


~
:0
[ram] 0:vi* 1:bas


23f3000648@cs1102: ~
#!/bin/bash
#Creating the directories
mkdir -p project/directory1
mkdir -p project/directory2
#Creating the files in their repsec


~
:5
[ram] 0:vi* 1:bas


23f3000648@cs1102: ~
#!/bin/bash
#Creating the directories
mkdir -p project/directory2
#Creating the files in their repsec
touch project/directory1/file1.txt
touch project/directory1/file2.txt
touch project/directory2/file3.txt
touch project/directory2/file4.txt


#Creating the files in their repsec
touch project/directory1/file1.txt
touch ram project/directory1/file2.
touch project/directory2/file3.txt
touch project/directory2/file4.txt

```
mkdir -p project/directory2

#Creating the files in their repsective direc
touch project/directory1/file1.txt
touch p what roject/directory1/file2.txt
touch project/directory2/file3.txt
touch project/directory2/file4.txt
```

```
mkdir -p project/directory2

#Creating the files in their repsective directori
touch project/directory1/file1.txt
touch project/directory2/file3.txt
touch project/directory2/file4.txt
```

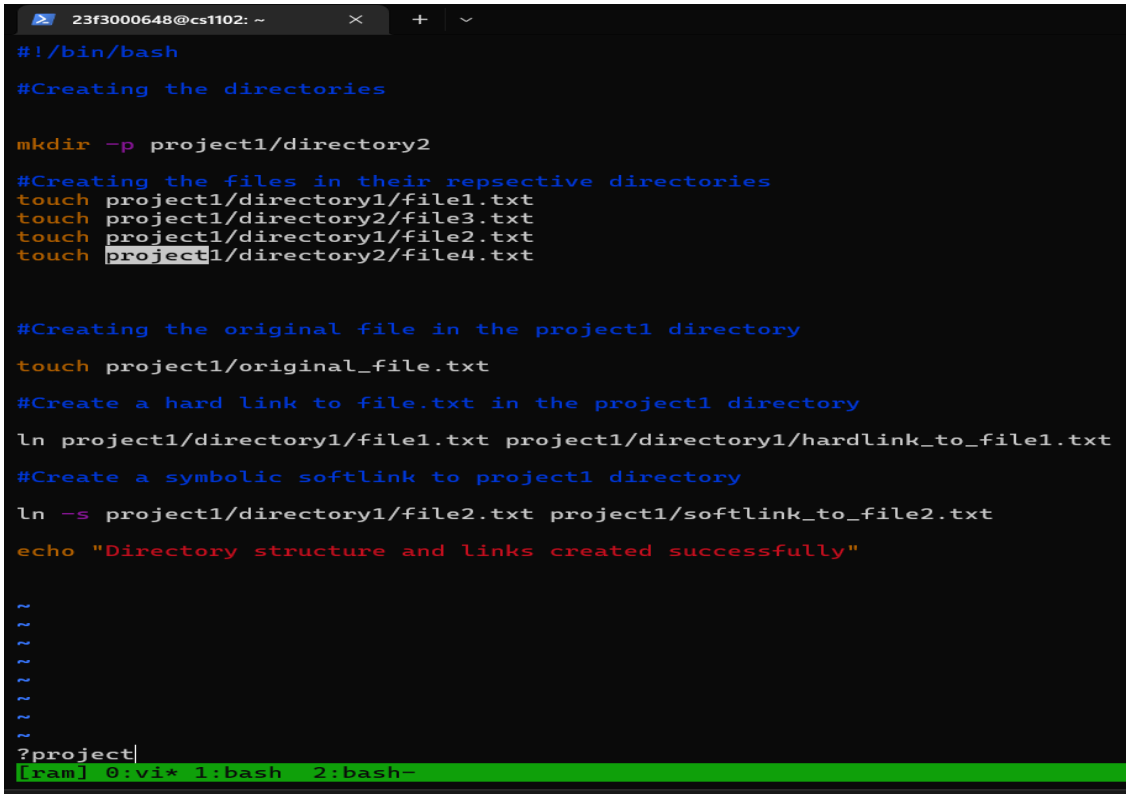
```
mkdir -p project/directory2

#Creating the files in their repsective dire
touch project/directory1/file1.txt
touch project/directory2/file3.txt
touch project/directory1/file2.txt
touch project/directory2/file4.txt

touch project/directory1/file2.txt

#Creating the original file in the project d

touch project/original_file.txt

#Create a hard link to file.txt in the proje
```

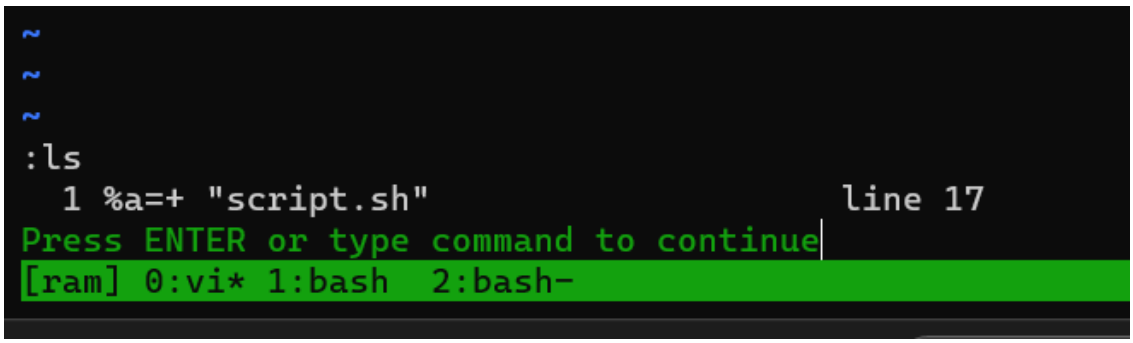
```
~
~
:%s/project/project1/g
[ram] 0:vi* 1:bash  2:bash-
```

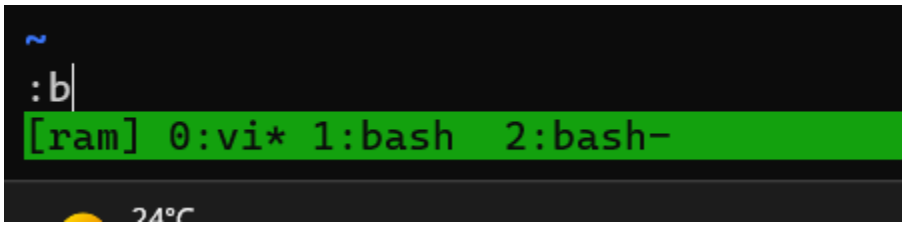
23f3000648@cs1102: ~

```
#!/bin/bash

#Creating the directories


mkdir -p project/directory2

#Creating the files in their repsective directories
touch project/directory1/file1.txt
touch project/directory2/file3.txt
touch project/directory1/file2.txt
touch project/directory2/file4.txt



#Creating the original file in the project directory

touch project/original_file.txt
```

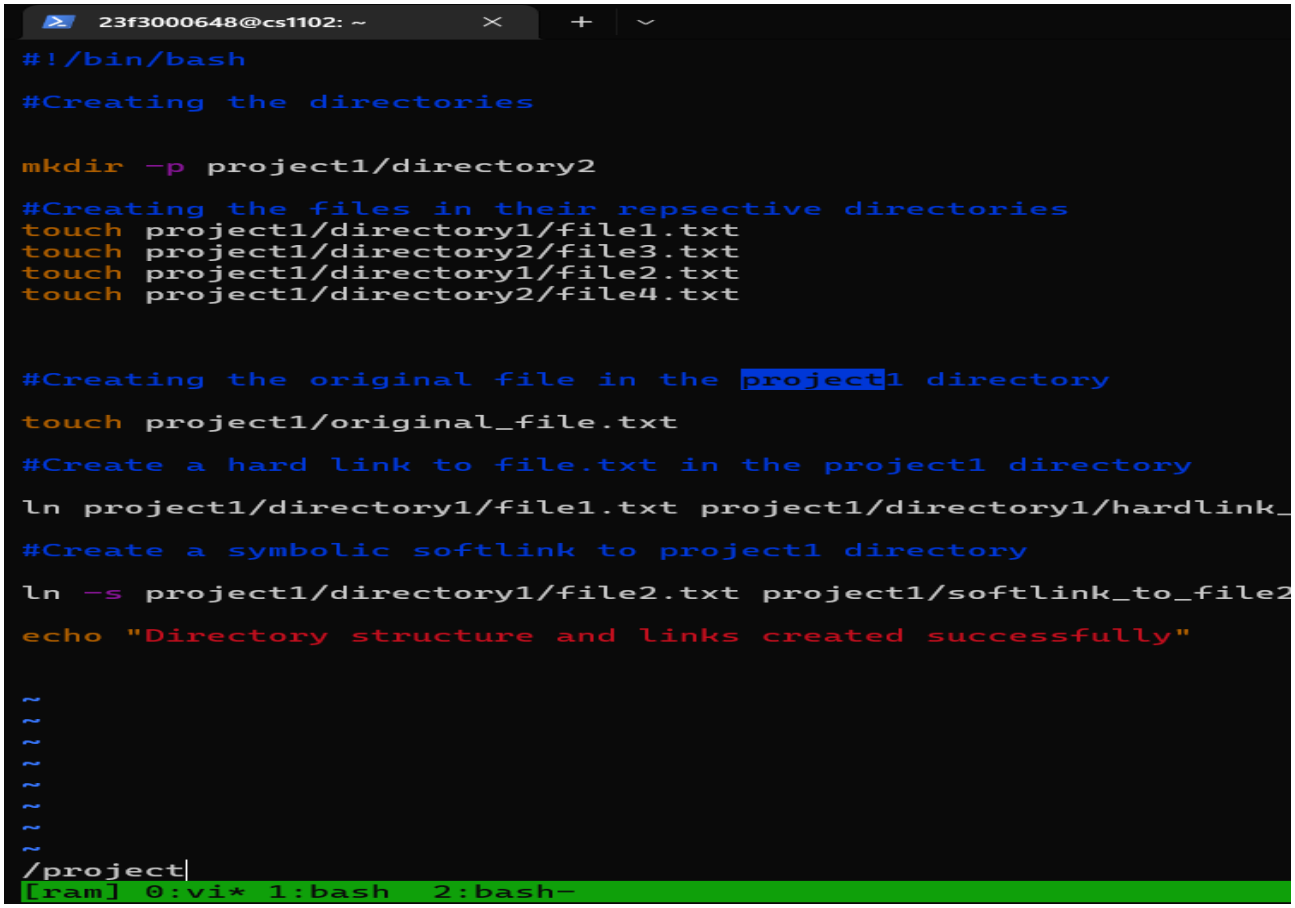
```
#Creating the directories


mkdir -p project1/directory2

#Creating the files in their repsective director
touch project1/directory1/file1.txt
touch project1/directory2/file3.txt
touch project1/directory1/file2.txt
touch project1/directory2/file4.txt



#Creating the original file in the project1 dire

touch project1/original_file.txt

#Create a hard link to file.txt in the project1
```


23f3000648@cs1102: ~
#!/bin/bash
#Creating the directories
mkdir -p project1/directory2
#Creating the files in their repsective directories
touch project1/directory1/file1.txt
touch project1/directory2/file3.txt
touch project1/directory1/file2.txt
touch project1/directory2/file4.txt
#Creating the original file in the project1 directory
touch project1/original_file.txt
#Create a hard link to file.txt in the project1 directory
ln project1/directory1/file1.txt project1/directory1/hardlink_to_file1.txt
#Create a symbolic softlink to project1 directory
ln -s project1/directory1/file2.txt project1/softlink_to_file2.txt
echo "Directory structure and links created successfully"
?project
[ram] 0:vi* 1:bash 2:bash-

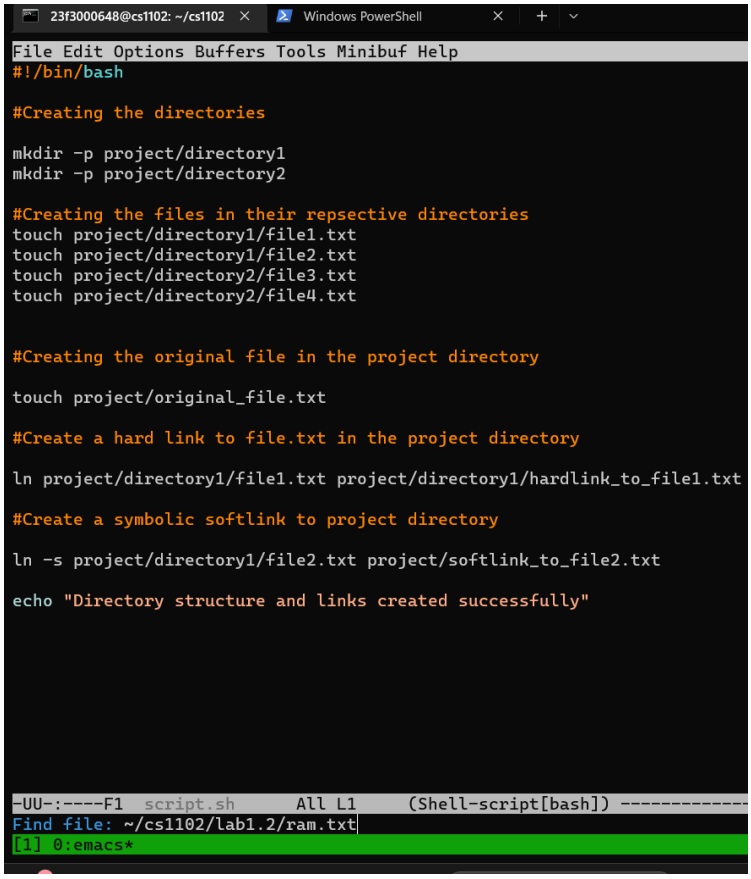

:ls
1 %a=+ "script.sh" line 17
Press ENTER or type command to continue
[ram] 0:vi* 1:bash 2:bash-

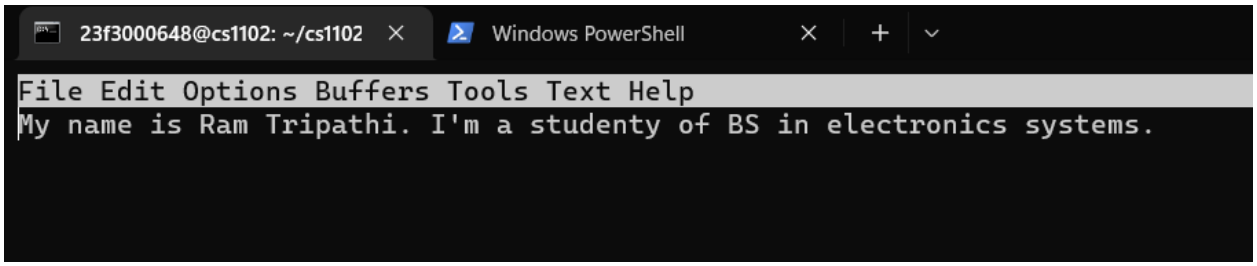

:b
[ram] 0:vi* 1:bash 2:bash-


echo "Directory structure and links created successfully"
script.sh [RO] 18,0-1 81%
:vsp ^W
[ram] 0:vi* 1:bash 2:bash-

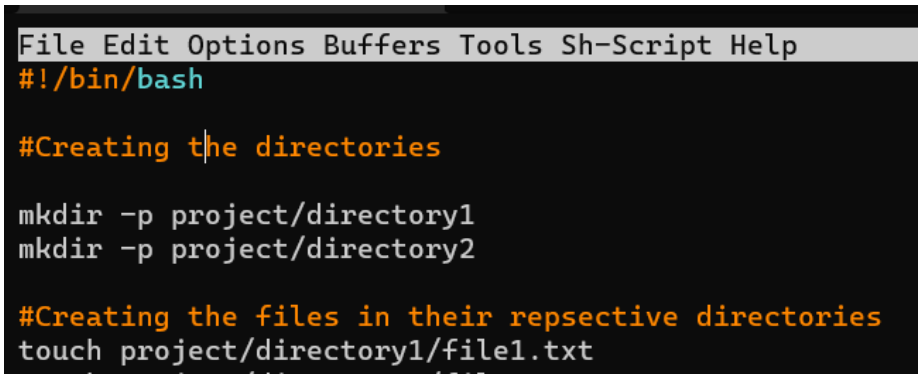


LAB:1.7 Emacs Editor

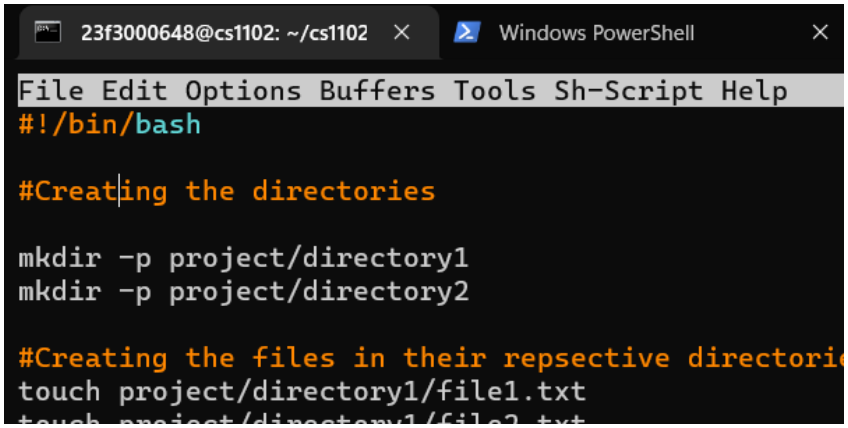




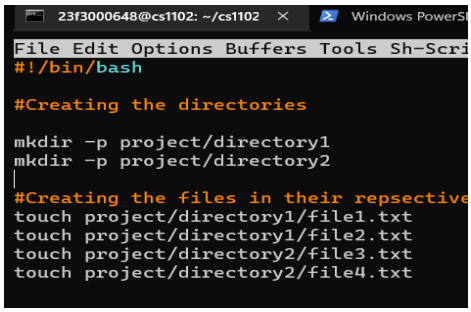
Moving forward using ctrl+f



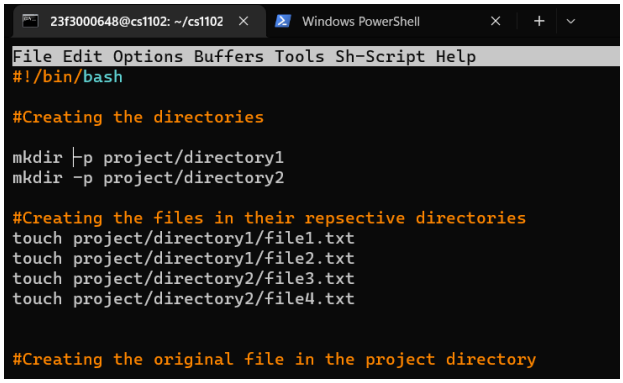
Moving backward using ctrl+b



Next line



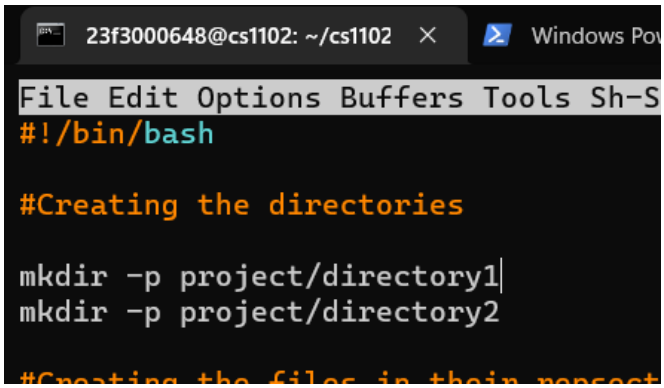
Previous line



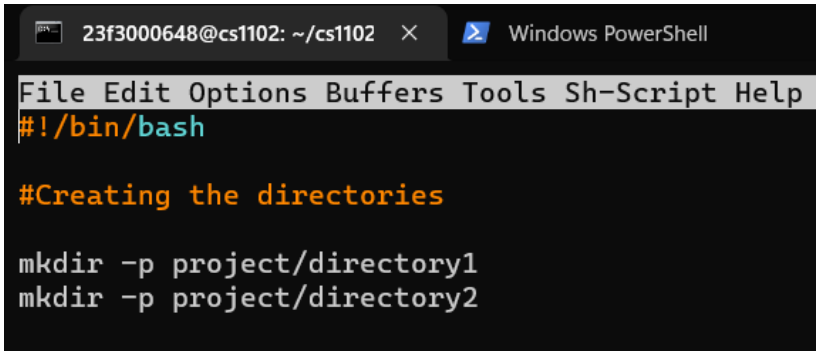
Beginning of line using ctrl+a

```
#!/bin/bash

#Creating the directories

mkdir -p project/directory1
mkdir -p project/directory2

#Creating the files in their reps
```

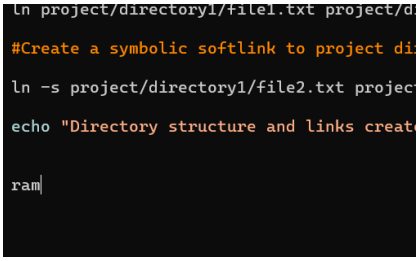
End of line using ctrl+e

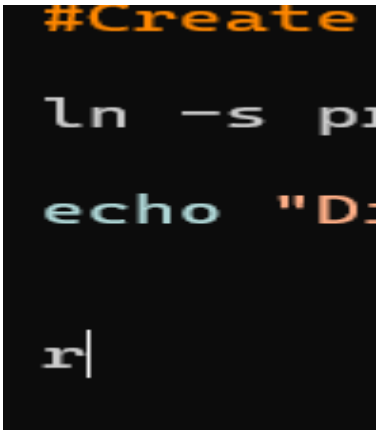


Beginning of buffer using ctrl+<



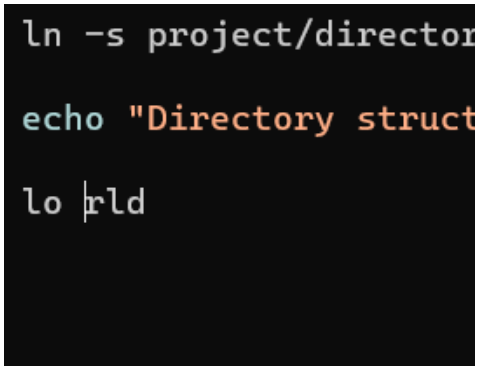
End of buffer using alt+>

Typing at current cursor position 

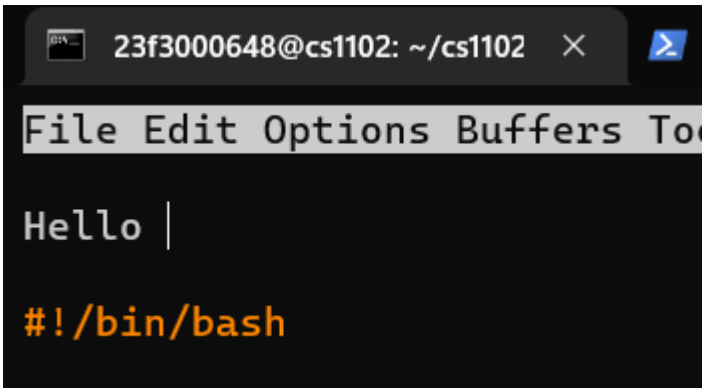
using back space to delete 

delete forward using ctrl+d

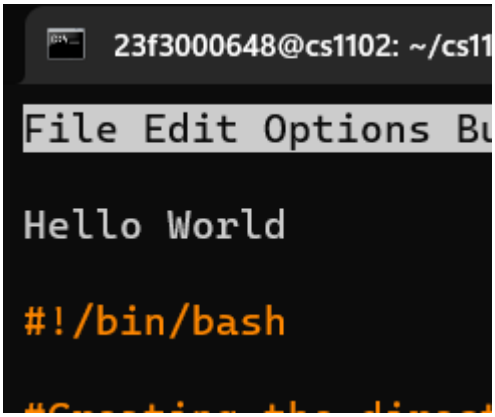
Delete backward using ctrl+h



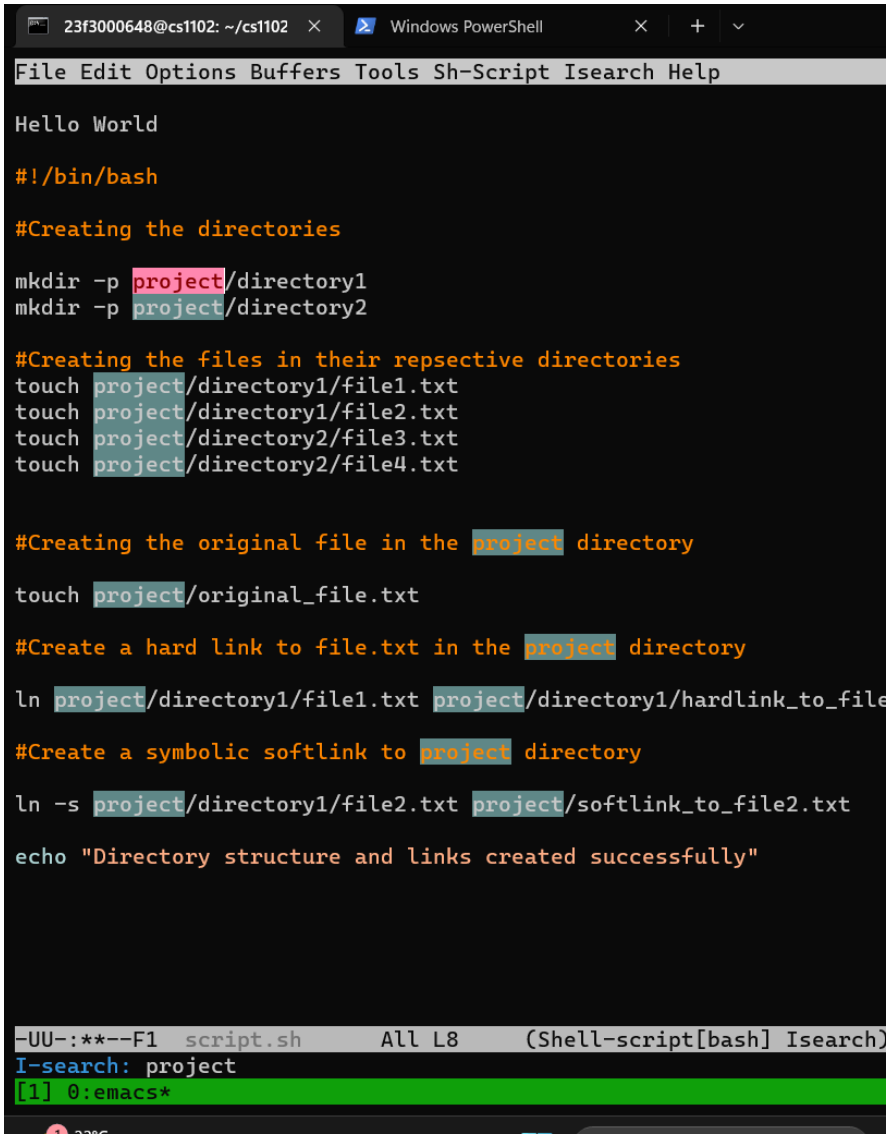
Cut using ctrl+k



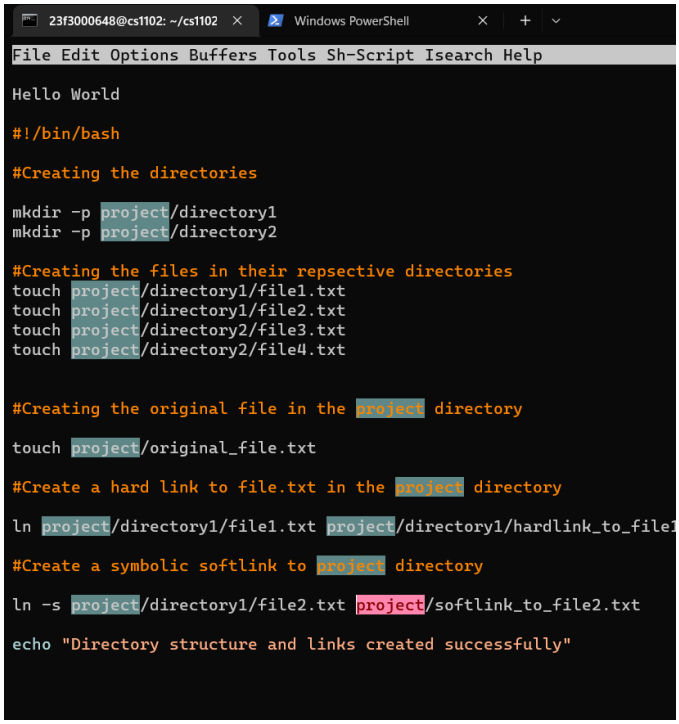
Paste using ctrl+y



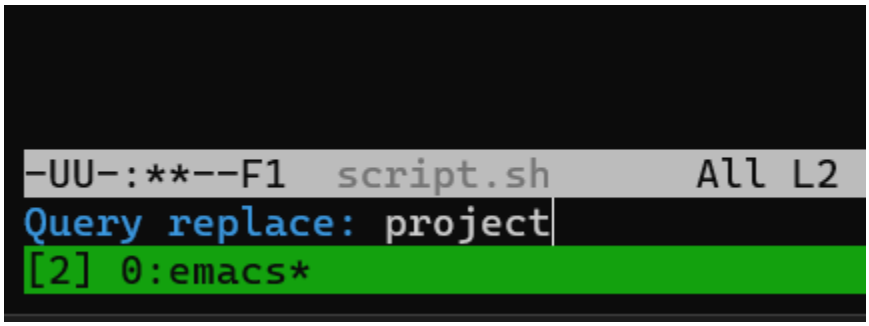
Search forward using ctrl+s

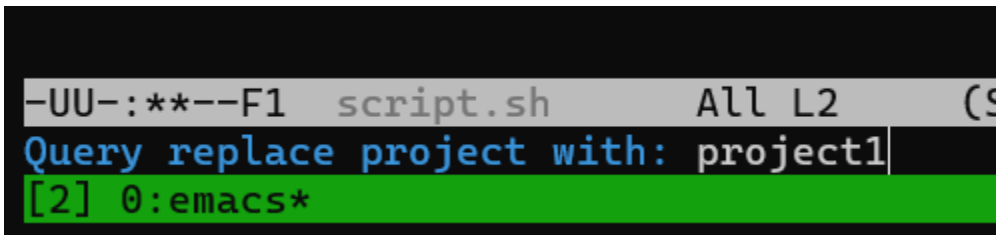


Search reverse with ctrl+r



Search and replace using Alt+%

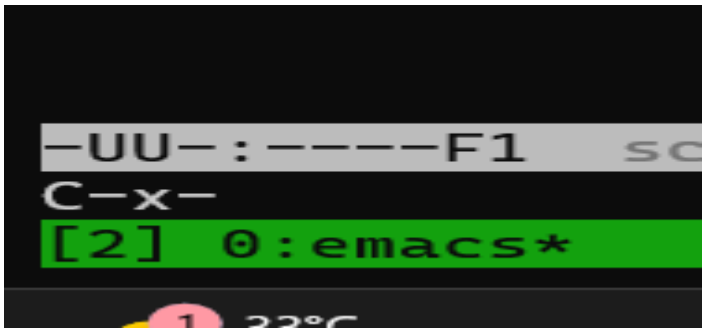




```
#Creating the files in their repsective directories
touch project1/directory1/file1.txt
touch project1/directory1/file2.txt
touch project1/directory2/file3.txt
touch project1/directory2/file4.txt


#Creating the original file in the project1 directory

touch project1/original_file.txt

#Create a hard link to file.txt in the project1 directory

ln project1/directory1/file1.txt project1/directory1/hardlin

#Create a symbolic softlink to project1 directory

ln -s project1/directory1/file2.txt project1/softlink_to_fil

echo "Directory structure and links created successfully"
```

```
-UU-:**--F1  script.sh      All L26    (Shell-script[bash])
Replaced 12 occurrences
[2] 0:emacs*
```

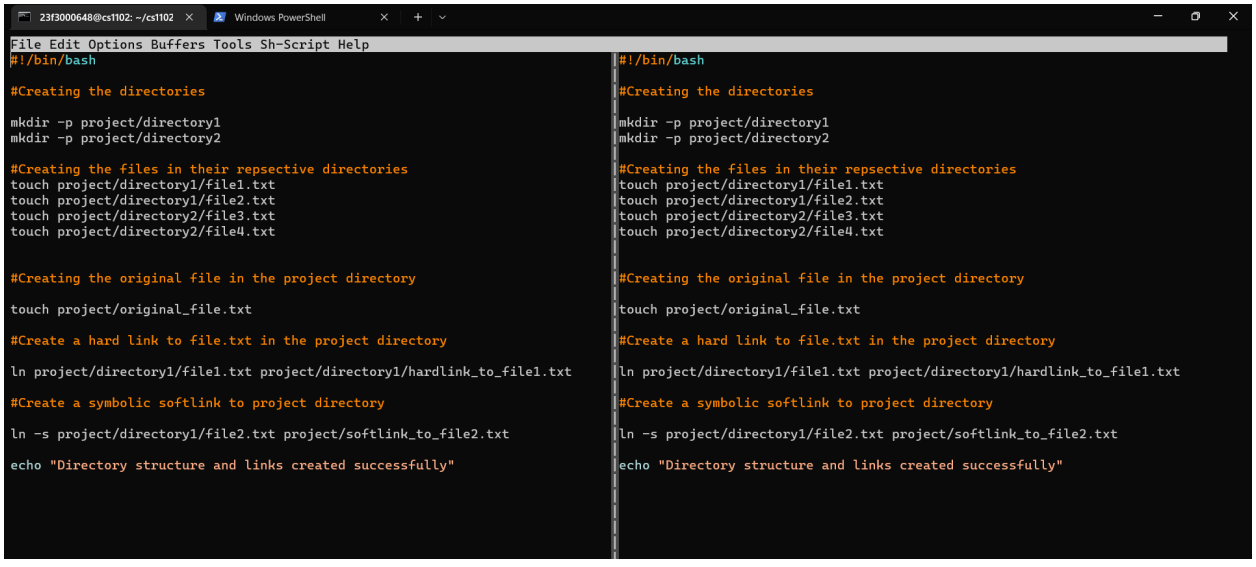
List buffers using ctrl+x ctrl+b



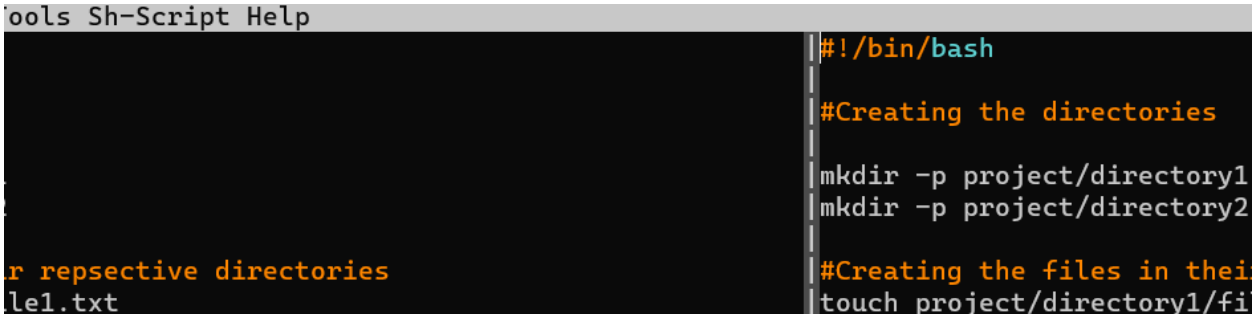
Split window vertically

```
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash

#Creating the directories

mkdir -p project/directory1
mkdir -p project/directory2

#Creating the files in their repsective directories
touch project/directory1/file1.txt
touch project/directory1/file2.txt
touch project/directory2/file3.txt
touch project/directory2/file4.txt


#Creating the original file in the project directory

touch project/original_file.txt

-UU-:----F1  script.sh      Top L1     (Shell-script[bash]) ------
#!/bin/bash

#Creating the directories

mkdir -p project/directory1
mkdir -p project/directory2

#Creating the files in their repsective directories
touch project/directory1/file1.txt
touch project/directory1/file2.txt
touch project/directory2/file3.txt
touch project/directory2/file4.txt


#Creating the original file in the project directory

touch project/original_file.txt
-UU-:----F1  script.sh      Top L1     (Shell-script[bash]) ------
```

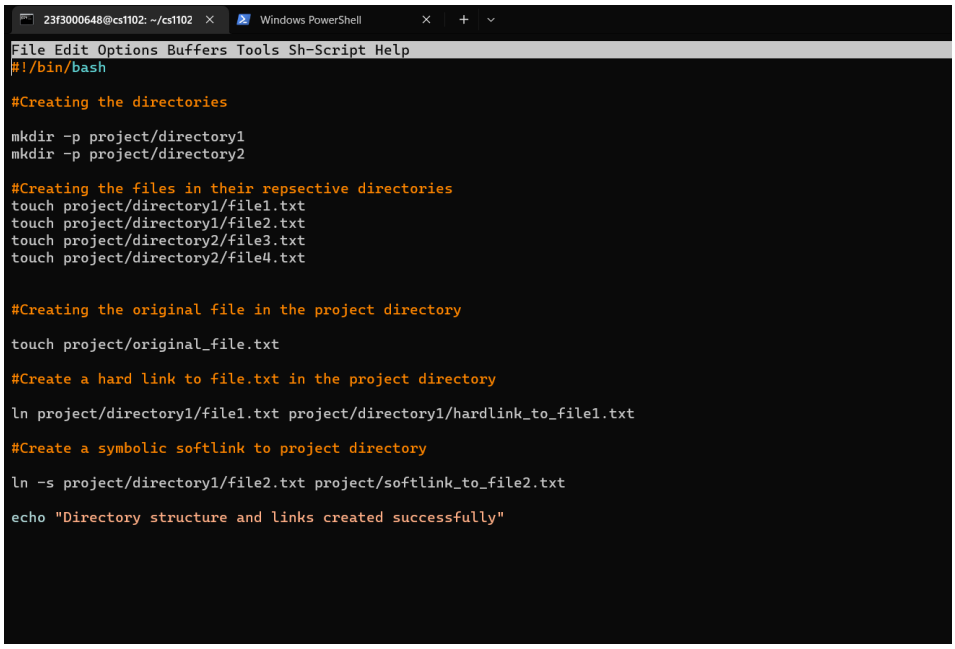
Split window vertically



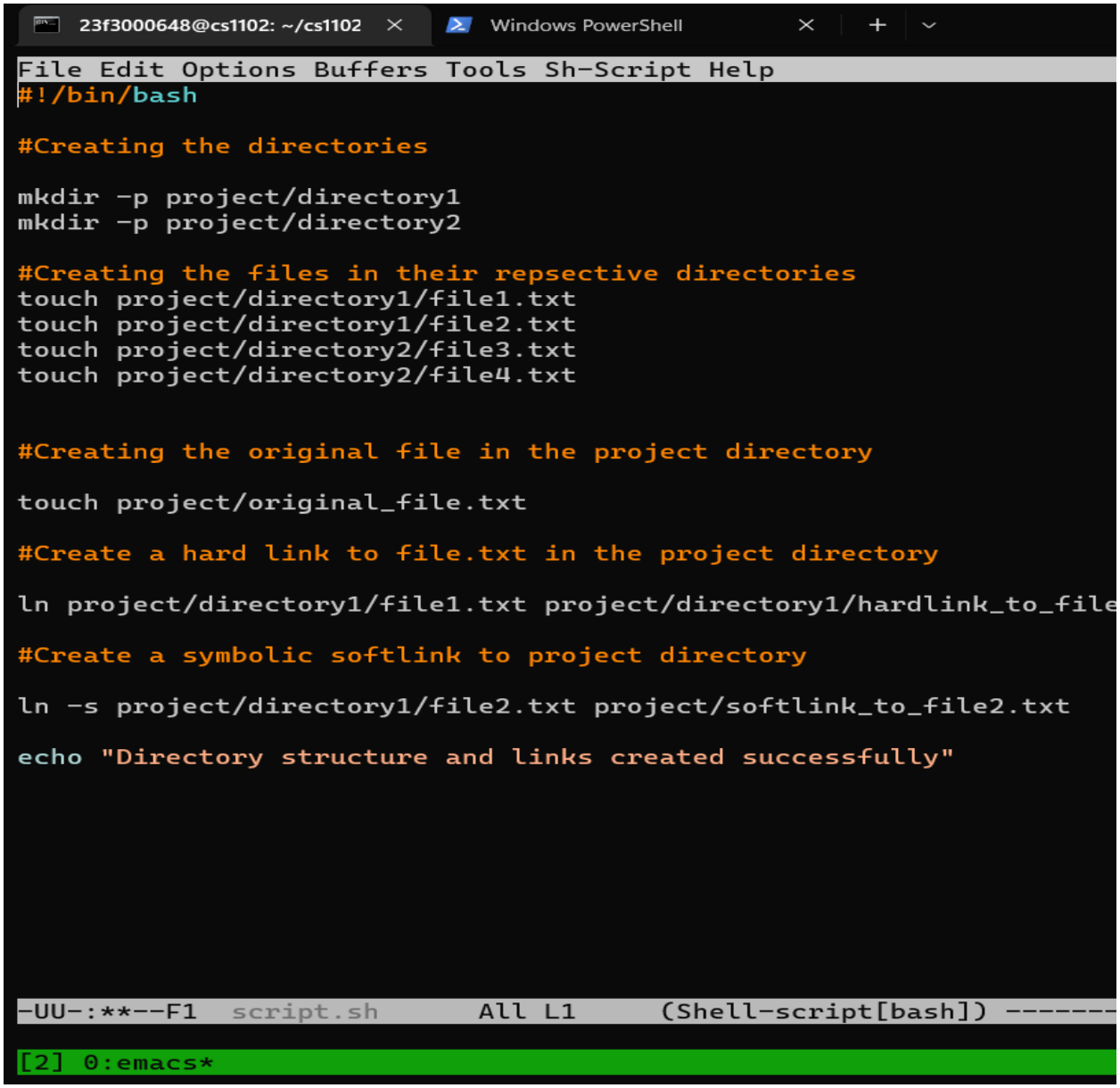
Switch between windows



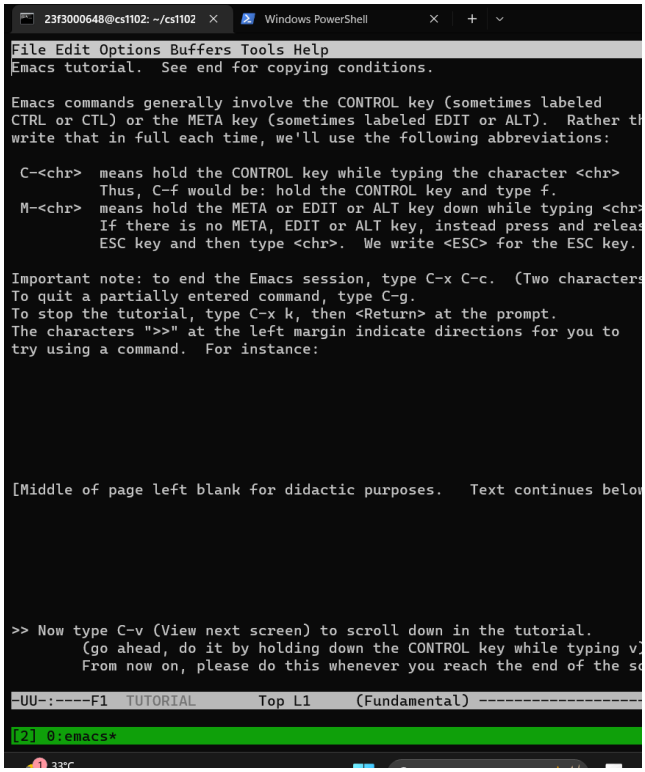
Closing a window



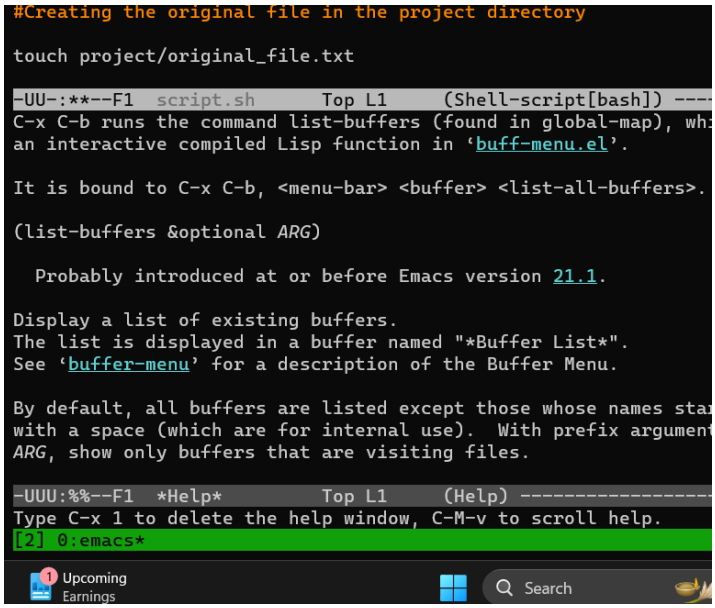
Close all windows except current one



EMACS TUTORIAL



Get help on a command using ctrl+h k



S ABULAMAN Kindly check for lab tasks from here onwards

Lab 2.1 Sleep Command Operations

Launch the sleep command with durations 1, 5, 100 and 1000 seconds

```
23f3000648@cs1102:~/cs1102/lab2.3$ sleep 1 5 100 1000
```

i. Send the sleep command to the background.

```
23f3000648@cs1102:~/cs1102/lab2.3$ sleep 1 5 100 1000
^Z
[1]+  Stopped                 sleep 1 5 100 1000
23f3000648@cs1102:~/cs1102/lab2.3$ bg
[1]+ sleep 1 5 100 1000 &
```

ii. Bring the sleep command to the foreground.

```
23f3000648@cs1102:~/cs1102/lab2.3$ fg
sleep 1 5 100 1000
```

iii. List the PID of the sleep command

```
23f3000648@cs1102:~/cs1102/lab2.3$ jobs
[1]+  Running                 sleep 1 5 100 1000 &
23f3000648@cs1102:~/cs1102/lab2.3$ jobs -p
538703
```

iv. Run the sleep command in both the shell and subshells.

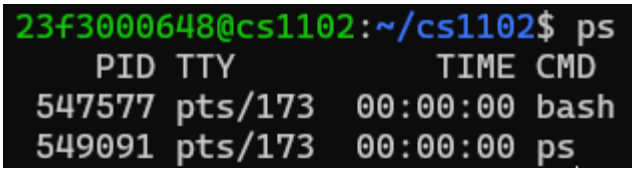
```
23f3000648@cs1102:~/cs1102/lab2.3$ sleep 1000 &
[1] 544114
23f3000648@cs1102:~/cs1102/lab2.3$ (sleep 999) &
[2] 544122
23f3000648@cs1102:~/cs1102/lab2.3$ jobs
[1]-  Running                 sleep 1000 &
[2]+  Running                 ( sleep 999 ) &
```

v. Demonstrate the use of pstree command, ps, ps -aux.

pstree : to visualize process hierarchy in Linux system

```
23f3000648@cs1102:~/cs1102$ pstree
systemd─┬─2*[agetty]
        ├─chronyd───chronyd
        ├─cron
        ├─dbus-daemon
        ├─google_guest_ag───9*[{google_guest_ag}]
        ├─google_osconfig───8*[{google_osconfig}]
        ├─multipathd───6*[{multipathd}]
        ├─networkd-dispat
        ├─nginx───2*[nginx]
        ├─packagekitd───2*[{packagekitd}]
        ├─polkitd───2*[{polkitd}]
        ├─rsyslogd───3*[{rsyslogd}]
        ├─2*[script───bash]
        ├─script───bash───script───bash───nano
        ├─snapd───8*[{snapd}]
        ├─sshd─┬─13*[sshd───sshd───bash]
        │      ├─2*[sshd───sshd───bash───emacs]
        │      ├─sshd───sshd───bash───batcat───less
        │      ├─sshd───sshd───bash───pstree
        │      ├─3*[sshd───sshd───bash───sleep]
        │      ├─sshd───sshd───bash───vi
        │      ├─2*[sshd]
        │      └─sshd───sshd
        ├─supervisord───deployment_scri───gunicorn───gunicorn
        ├─113*[systemd───(sd-pam)]
        ├─systemd-journal
        ├─systemd-logind
        ├─systemd-network
        ├─systemd-resolve
        ├─systemd-udevd
        ├─2*[tmux: server───7*[bash]]
        ├─3*[tmux: server─┬─2*[bash]]
        │                 └─bash───vi]
        ├─tmux: server─┬─4*[bash]
        │              └─3*[bash───vi]
        ├─tmux: server─┬─14*[bash]
        │              ├─bash───bash───batcat───less
        │              └─2*[bash───info]
        ├─4*[tmux: server───6*[bash]]
        ├─2*[tmux: server─┬─2*[bash]]
        │                 └─bash───emacs]
        ├─tmux: server─┬─10*[bash]
        │              └─11*[bash───vi]
        ├─3*[tmux: server─┬─bash]
        │                 └─bash───vi]
        └─26*[tmux: server───bash]
```

ps : to report snapshot of the current processes (bound by the controlling terminal)



ps -aux : the -aux option is grouping of three individual options namely:

-a = show processes for all users

-u = display the process's user/owner

-x = also show processes not attached to a terminal

```
23f3000648@cs1102:~/cs1102/lab2.1$ ps -aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.3 171436 14196 ?        Ss   Oct09   4:10 /sbin/ini
root           2  0.0  0.0      0     0 ?        S    Oct09   0:00 [kthreadd
root           3  0.0  0.0      0     0 ?        I<   Oct09   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   Oct09   0:00 [rcu_par_
root           5  0.0  0.0      0     0 ?        I<   Oct09   0:00 [slub_flu
root           6  0.0  0.0      0     0 ?        I<   Oct09   0:00 [netns]
root           8  0.0  0.0      0     0 ?        I<   Oct09   0:00 [kworker/
root          10  0.0  0.0      0     0 ?        I<   Oct09   0:00 [mm_percp
root          11  0.0  0.0      0     0 ?        I    Oct09   0:00 [rcu_task
root          12  0.0  0.0      0     0 ?        I    Oct09   0:00 [rcu_task
root          13  0.0  0.0      0     0 ?        S    Oct09   0:44 [ksoftirq
root          14  0.0  0.0      0     0 ?        I    Oct09   1:01 [rcu_sche
root          15  0.0  0.0      0     0 ?        S    Oct09   0:08 [migratio
root          16  0.0  0.0      0     0 ?        S    Oct09   0:00 [idle_inj
root          18  0.0  0.0      0     0 ?        S    Oct09   0:00 [cpuhp/0]
root          19  0.0  0.0      0     0 ?        S    Oct09   0:00 [cpuhp/1]
root          20  0.0  0.0      0     0 ?        S    Oct09   0:00 [idle_inj
root          21  0.0  0.0      0     0 ?        S    Oct09   0:08 [migratio
root          22  0.0  0.0      0     0 ?        S    Oct09   0:45 [ksoftirq
root          24  0.0  0.0      0     0 ?        I<   Oct09   0:00 [kworker/
root          25  0.0  0.0      0     0 ?        S    Oct09   0:00 [kdevtmpf
root          26  0.0  0.0      0     0 ?        I<   Oct09   0:00 [inet_fra
root          27  0.0  0.0      0     0 ?        S    Oct09   0:00 [kauditd]
root          29  0.0  0.0      0     0 ?        S    Oct09   0:03 [khungtas
root          31  0.0  0.0      0     0 ?        S    Oct09   0:00 [oom_reap
root          32  0.0  0.0      0     0 ?        I<   Oct09   0:00 [writebac
root          34  0.0  0.0      0     0 ?        S    Oct09   1:27 [kcompact
root          35  0.0  0.0      0     0 ?        SN   Oct09   0:00 [ksmd]
root          36  0.0  0.0      0     0 ?        SN   Oct09   0:00 [khugepag
root          37  0.0  0.0      0     0 ?        I<   Oct09   0:00 [kintegri
root          38  0.0  0.0      0     0 ?        I<   Oct09   0:00 [kblockd]
root          39  0.0  0.0      0     0 ?        I<   Oct09   0:00 [blkcg_pu
```

vi. Schedule a sleep command at a predefined time, day, and date.

```
23f3000648@cs1102:~/cs1102/lab2.1$ echo "sleep 100" | at 10:45 tomorrow
warning: commands will be executed using /bin/sh
job 31 at Thu Oct 24 10:45:00 2024
```

```
23f3000648@cs1102:~/cs1102/lab2.1$ at -l
31      Thu Oct 24 10:45:00 2024 a 23f3000648
```

vii. Halt the sleep commands

```
23f3000648@cs1102:~/cs1102/lab2.1$ sleep 1000 &
[1] 560914
23f3000648@cs1102:~/cs1102/lab2.1$ kill -SIGSTOP 560914

[1]+  Stopped                 sleep 1000
23f3000648@cs1102:~/cs1102/lab2.1$ jobs
[1]+  Stopped                 sleep 1000
```

Alternatively:

```
23f3000648@cs1102:~/cs1102/lab2.3$ sleep 1000
^Z
[1]+  Stopped                 sleep 1000
```

viii. Interrupt the sleep commands.

```
23f3000648@cs1102:~/cs1102/lab2.1$ sleep 1000 &
[1] 560075
23f3000648@cs1102:~/cs1102/lab2.1$ kill -SIGINT 560075
23f3000648@cs1102:~/cs1102/lab2.1$ jobs
[1]+  Interrupt               sleep 1000
```

ix. Kill the sleep commands.

```
23f3000648@cs1102:~/cs1102/lab2.1$ sleep 1000 &
[1] 560489
23f3000648@cs1102:~/cs1102/lab2.1$ kill -SIGKILL 560489
23f3000648@cs1102:~/cs1102/lab2.1$ jobs
[1]+  Killed                  sleep 1000
```

2.2 Use Cases of cat Command

Display file contents

```
23f3000648@cs1102:~/cs1102/lab2.2$ ls
file1.txt
23f3000648@cs1102:~/cs1102/lab2.2$ cat file1.txt
Randomly typed contents of file called file1.txt
```

Display contents of multiple files
(concatenates the file contents one after the other)

```
23f3000648@cs1102:~/cs1102/lab2.2$ ls
file1.txt  file2.txt
23f3000648@cs1102:~/cs1102/lab2.2$ cat file1.txt
Randomly typed contents of file called file1.txt
23f3000648@cs1102:~/cs1102/lab2.2$ cat file2.txt
Randomly typed contents of file called file2.txt
23f3000648@cs1102:~/cs1102/lab2.2$ cat file1.txt file2.txt
Randomly typed contents of file called file1.txt
Randomly typed contents of file called file2.txt
```

Creating a file and writing to it at the same time

```
23f3000648@cs1102:~/cs1102/lab2.2$ ls
file1.txt  file2.txt
23f3000648@cs1102:~/cs1102/lab2.2$ cat > fileCreatedByCat.txt
This file was created using cat.
23f3000648@cs1102:~/cs1102/lab2.2$ ls
file1.txt  file2.txt  fileCreatedByCat.txt
23f3000648@cs1102:~/cs1102/lab2.2$ cat fileCreatedByCat.txt
This file was created using cat.
23f3000648@cs1102:~/cs1102/lab2.2$
```

Appending to the above created file using cat

```
23f3000648@cs1102:~/cs1102/lab2.2$ ls
file1.txt  file2.txt  fileCreatedByCat.txt
23f3000648@cs1102:~/cs1102/lab2.2$ cat >> fileCreatedByCat.txt
This was appended using cat.
23f3000648@cs1102:~/cs1102/lab2.2$ cat fileCreatedByCat.txt
This file was created using cat.
This was appended using cat.
```

Concatenate multiple files using cat

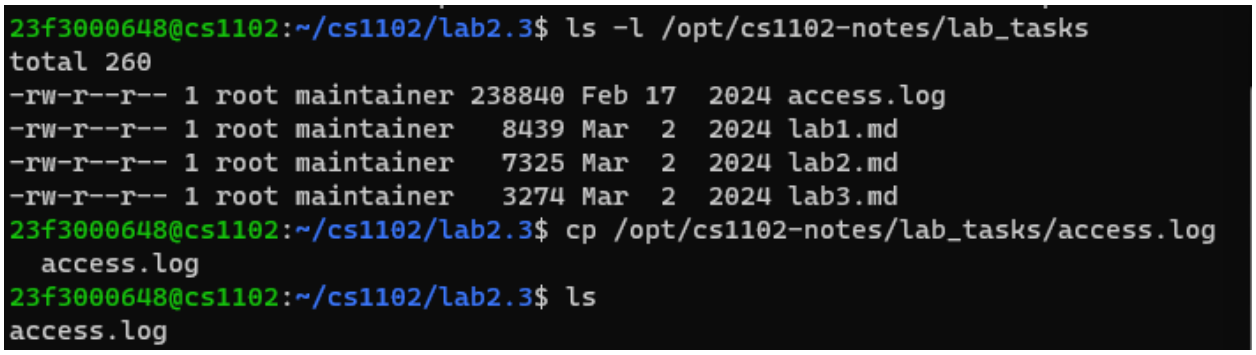
```
23f3000648@cs1102:~/cs1102/lab2.2$ ls
file1.txt  file2.txt  fileCreatedByCat.txt
23f3000648@cs1102:~/cs1102/lab2.2$ cat fileCreatedByCat.txt
This file was created using cat.
This was appended using cat.
23f3000648@cs1102:~/cs1102/lab2.2$ cat file1.txt
Randomly typed contents of file called file1.txt
23f3000648@cs1102:~/cs1102/lab2.2$ cat file2.txt
Randomly typed contents of file called file2.txt
23f3000648@cs1102:~/cs1102/lab2.2$ cat file1.txt file2.txt >> fileCreatedByCat.txt
23f3000648@cs1102:~/cs1102/lab2.2$ cat fileCreatedByCat.txt
This file was created using cat.
This was appended using cat.
Randomly typed contents of file called file1.txt
Randomly typed contents of file called file2.txt
```

Copy file content

```
23f3000648@cs1102:~/cs1102/lab2.2$ ls
file1.txt  file2.txt  fileCreatedByCat.txt
23f3000648@cs1102:~/cs1102/lab2.2$ cat fileCreatedByCat.txt > copy.txt
23f3000648@cs1102:~/cs1102/lab2.2$ cat fileCreatedByCat.txt
This file was created using cat.
This was appended using cat.
Randomly typed contents of file called file1.txt
Randomly typed contents of file called file2.txt
23f3000648@cs1102:~/cs1102/lab2.2$ ls
copy.txt  file1.txt  file2.txt  fileCreatedByCat.txt
23f3000648@cs1102:~/cs1102/lab2.2$ cat copy.txt
This file was created using cat.
This was appended using cat.
Randomly typed contents of file called file1.txt
Randomly typed contents of file called file2.txt
```

2.3 Processing access.log

Creating a local copy of access.log in ~/cs1102/lab2.3



i. Create a list of unique IP addresses present in the log

```
23f3000648@cs1102:~/cs1102/lab2.3$ grep "([[:digit:]]{1,3}\.){3}[[:digit:]]{1,3}" access.log -oE | uniq -u | sort
103.0.0.0
103.158.43.33
103.224.35.184
106.51.180.232
106.51.180.232
107.178.231.229
107.178.231.229
108.0.0.0
108.0.0.0
108.0.0.0
108.0.0.0
109.0.0.0
109.0.0.0
109.0.0.0
109.0.0.0
11.0.5.850
11.0.5.850
11.0.5.850
11.0.5.850
110.0.0.0
110.0.0.0
112.0.0.0
112.0.0.0
112.0.0.0
112.0.0.0
114.0.0.0
114.0.0.0
117.233.197.219
121.0.0.0
121.0.0.0
121.0.0.0
121.0.0.0
121.0.0.0
141.98.11.89
141.98.11.89
141.98.11.89
141.98.11.89
141.98.11.89
```

ii. Print the total count of entries encountering 404 error

```
23f3000648@cs1102:~/cs1102/lab2.3$ grep -w "404" access.log | wc -l
435
```

iii. Extract only the time data from the log, keeping only hours and minutes

```
23f3000648@cs1102:~/cs1102/lab2.3$ grep -oE "([0-9]{2}:){2}[0-9]{2}" access.log
24:00:11
24:00:13
24:00:16
24:00:20
24:00:24
24:00:28
24:00:28
24:00:29
24:00:29
24:00:29
24:00:29
24:00:29
24:00:29
24:00:29
24:00:29
24:00:29
24:00:29
24:00:29
24:00:30
24:00:30
24:00:30
24:00:30
24:00:30
24:00:30
24:00:30
24:00:46
24:00:50
24:00:53
24:00:54
24:01:08
24:01:12
24:01:20
24:01:26
24:01:28
24:01:28
24:01:31
24:01:50
24:01:50
```

2.4

Creating a local copy of access.log in ~/cs1102/lab2.4

```
23f3000648@cs1102:~/cs1102/lab2.4$ cp ../lab2.3/access.log access.log
23f3000648@cs1102:~/cs1102/lab2.4$ ls
access.log
```

i. Search for all appearances of “Macintosh”

```
23f3000648@cs1102:~/cs1102/lab2.4$ vi access.log
```

Vi/Vim

```
83.97.73.245 - - [16/Feb/2024:00:11:33 +0000] "GET /?XDEBUG_SESSION_START=ph
pstorm HTTP/1.1" 404 184 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) Appl
eWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36"
157.245.176.143 - - [16/Feb/2024:00:13:54 +0000] "SSTP_DUPLEX_POST /sra_{BA1
95980-CD49-458b-9E23-C84EE0ADCD75}/ HTTP/1.1" 400 150 "-" "-"
45.79.163.53 - - [16/Feb/2024:00:16:06 +0000] "GET / HTTP/1.1" 400 248 "-" "
Mozilla/5.0 zgrab/0.x"
185.180.143.189 - - [16/Feb/2024:00:20:59 +0000] "GET / HTTP/1.1" 403 180 "-
" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/60.0.3112.113 Safari/537.36"
194.67.201.24 - - [16/Feb/2024:00:24:32 +0000] "GET / HTTP/1.1" 404 184 "-"
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_1) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/76.0.3809.100 Safari/537.36"
143.110.222.166 - - [16/Feb/2024:00:28:25 +0000] "GET / HTTP/1.1" 403 118 "-
" "Mozilla/5.0 (iPhone; CPU iPhone OS 16_1 like Mac OS X) AppleWebKit/605.1.
15 (KHTML, like Gecko) Version/16.1 Mobile/15E148 Safari/604.1"
45.79.181.94 - - [16/Feb/2024:00:28:58 +0000] "GET / HTTP/1.1" 404 184 "-" "
Mozilla/5.0 (Macintosh; Intel Mac OS X 13_1) AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/108.0.0.0 Safari/537.36"
161.35.27.144 - - [16/Feb/2024:00:29:48 +0000] "GET / HTTP/1.1" 404 207 "-"
"-"
161.35.27.144 - - [16/Feb/2024:00:29:53 +0000] "GET / HTTP/1.1" 404 207 "-"
"Mozilla/5.0 (Linux; Android 6.0; HTC One M9 Build/MRA591839) AppleWebKit/53
7.36 (KHTML, like Gecko) Chrome/52.0.3267.98 Mobile Safari/537.3"
161.35.27.144 - - [16/Feb/2024:00:29:54 +0000] "GET /version HTTP/1.1" 404 1
84 "-" "Go-http-client/1.1"
161.35.27.144 - - [16/Feb/2024:00:29:55 +0000] "GET /.vscode/sftp.json HTTP/
1.1" 404 184 "-" "Go-http-client/1.1"
161.35.27.144 - - [16/Feb/2024:00:29:55 +0000] "GET /about HTTP/1.1" 404 184
 "-" "Go-http-client/1.1"
161.35.27.144 - - [16/Feb/2024:00:29:56 +0000] "GET /debug/default/view?pane
l=config HTTP/1.1" 404 184 "-" "Go-http-client/1.1"
161.35.27.144 - - [16/Feb/2024:00:29:56 +0000] "GET /v2/_catalog HTTP/1.1" 4
04 184 "-" "Go-http-client/1.1"
161.35.27.144 - - [16/Feb/2024:00:29:57 +0000] "GET /ecp/Current/exporttool/
microsoft.exchange.ediscovery.exporttool.application HTTP/1.1" 404 184 "-" "
Go-http-client/1.1"
161.35.27.144 - - [16/Feb/2024:00:29:58 +0000] "GET /server-status HTTP/1.1"
 404 184 "-" "Go-http-client/1.1"
@@@
/Macintosh
```

Emacs

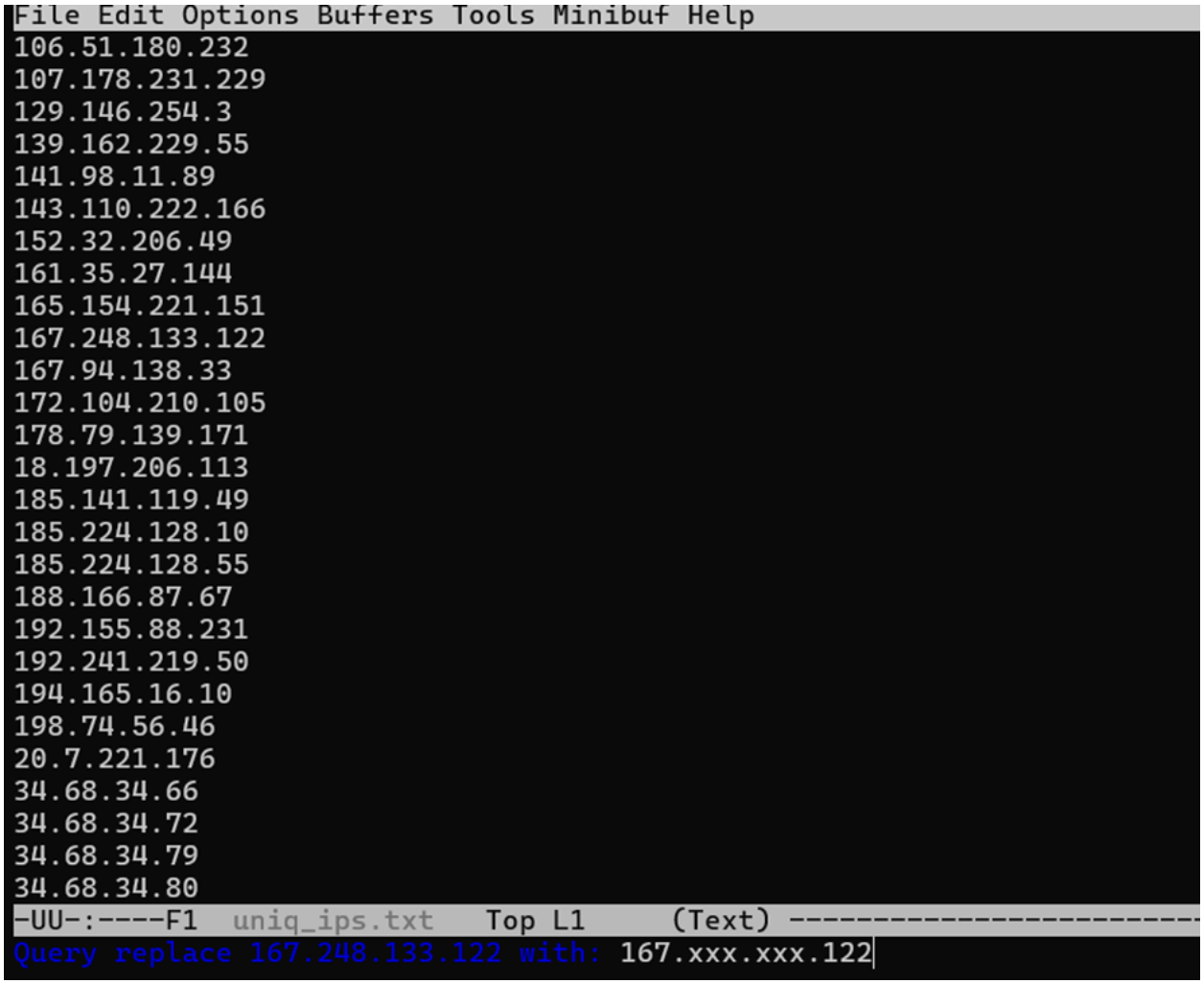
ii. Search and replace the IP address, masking 123.45.67.89 with 123.xx.xx.xx.89

Vi/Vim

+

```
83.xx.xx.245 - - [16/Feb/2024:00:11:33 +0000] "GET /?XDEBUG_SESSION_START=ph
pstorm HTTP/1.1" 404 184 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) Appl
eWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36"
157.xx.xx.143 - - [16/Feb/2024:00:13:54 +0000] "SSTP_DUPLEX_POST /sra_{BA195
980-CD49-458b-9E23-C84EE0ADCD75}/ HTTP/1.1" 400 150 "-" "-"
45.xx.xx.53 - - [16/Feb/2024:00:16:06 +0000] "GET / HTTP/1.1" 400 248 "-" "M
ozilla/5.0 zgrab/0.x"
185.xx.xx.189 - - [16/Feb/2024:00:20:59 +0000] "GET / HTTP/1.1" 403 180 "-"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like G
ecko) Chrome/60.0.3112.113 Safari/537.36"
194.xx.xx.24 - - [16/Feb/2024:00:24:32 +0000] "GET / HTTP/1.1" 404 184 "-" "
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_1) AppleWebKit/537.36 (KHTML, l
ike Gecko) Chrome/76.0.3809.100 Safari/537.36"
143.xx.xx.166 - - [16/Feb/2024:00:28:25 +0000] "GET / HTTP/1.1" 403 118 "-"
"Mozilla/5.0 (iPhone; CPU iPhone OS 16_1 like Mac OS X) AppleWebKit/605.1.15
 (KHTML, like Gecko) Version/16.1 Mobile/15E148 Safari/604.1"
45.xx.xx.94 - - [16/Feb/2024:00:28:58 +0000] "GET / HTTP/1.1" 404 184 "-" "M
ozilla/5.0 (Macintosh; Intel Mac OS X 13_1) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/108.xx.xx.0 Safari/537.36"
161.xx.xx.144 - - [16/Feb/2024:00:29:48 +0000] "GET / HTTP/1.1" 404 207 "-"
"-"
161.xx.xx.144 - - [16/Feb/2024:00:29:53 +0000] "GET / HTTP/1.1" 404 207 "-"
"Mozilla/5.0 (Linux; Android 6.0; HTC One M9 Build/MRA591839) AppleWebKit/53
7.36 (KHTML, like Gecko) Chrome/52.0.3267.98 Mobile Safari/537.3"
161.xx.xx.144 - - [16/Feb/2024:00:29:54 +0000] "GET /version HTTP/1.1" 404 1
84 "-" "Go-http-client/1.1"
161.xx.xx.144 - - [16/Feb/2024:00:29:55 +0000] "GET /.vscode/sftp.json HTTP/
1.1" 404 184 "-" "Go-http-client/1.1"
161.xx.xx.144 - - [16/Feb/2024:00:29:55 +0000] "GET /about HTTP/1.1" 404 184
 "-" "Go-http-client/1.1"
161.xx.xx.144 - - [16/Feb/2024:00:29:56 +0000] "GET /debug/default/view?pane
l=config HTTP/1.1" 404 184 "-" "Go-http-client/1.1"
161.xx.xx.144 - - [16/Feb/2024:00:29:56 +0000] "GET /v2/_catalog HTTP/1.1" 4
04 184 "-" "Go-http-client/1.1"
161.xx.xx.144 - - [16/Feb/2024:00:29:57 +0000] "GET /ecp/Current/exporttool/
microsoft.exchange.ediscovery.exporttool.application HTTP/1.1" 404 184 "-" "
Go-http-client/1.1"
161.xx.xx.144 - - [16/Feb/2024:00:29:58 +0000] "GET /server-status HTTP/1.1"
 404 184 "-" "Go-http-client/1.1"
@@@
:%s/\v\.([0-9]{1,3}\.){2}/\.\x\x\.\x\x\./g
```

Emacs



S ABULAMAN sir please evaluate henceforth

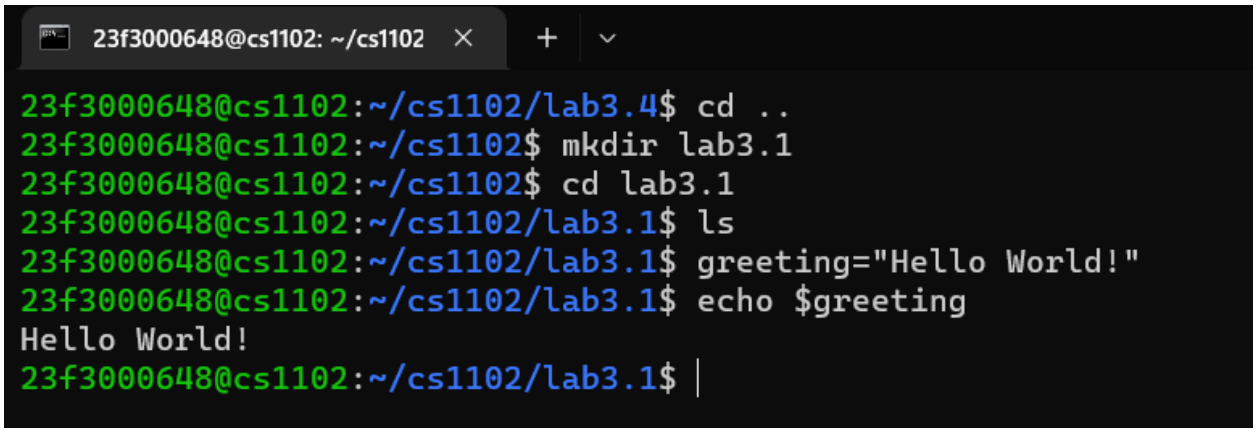
Lab 3 Activities

3.1 Variables

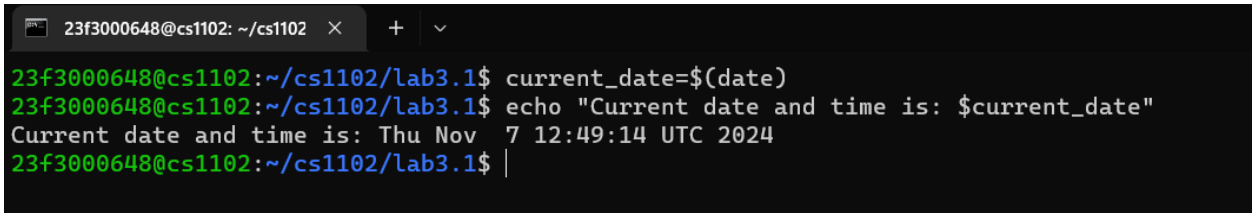
3.1.1 Assigning Values to a Variable

Demonstrate four use cases of assigning values to variables.

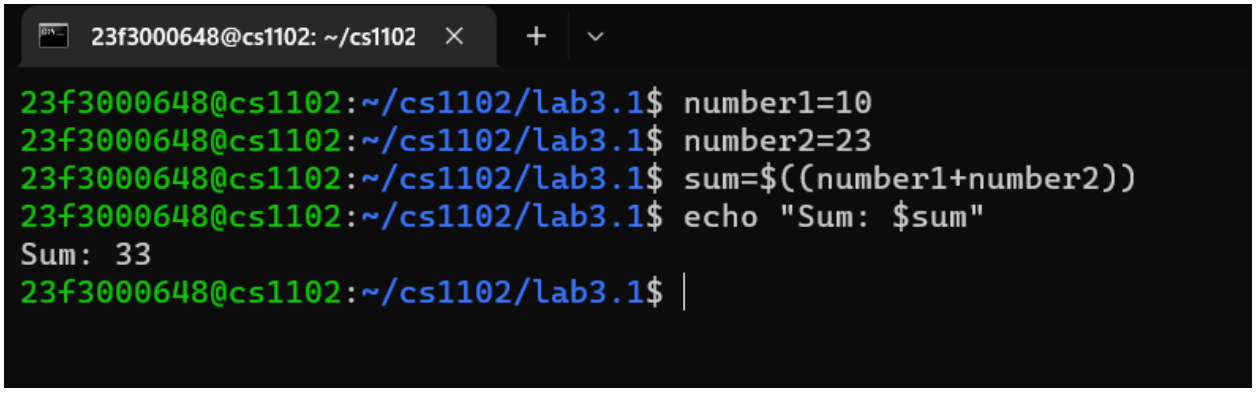
1. Assigning a String Value



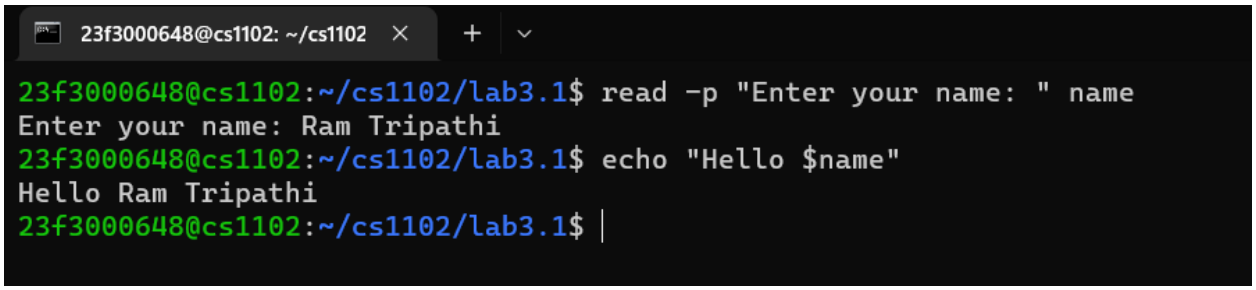
2. Assigning the Output of a Command



3. Assigning the Result of an Arithmetic Expression



4. Assigning a Value from User Input

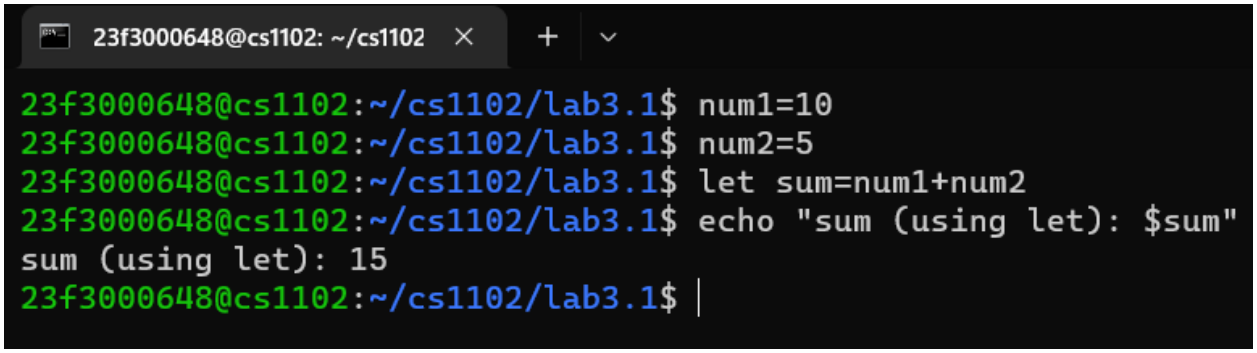


3.1.2 Bash Arithmetic with Variables

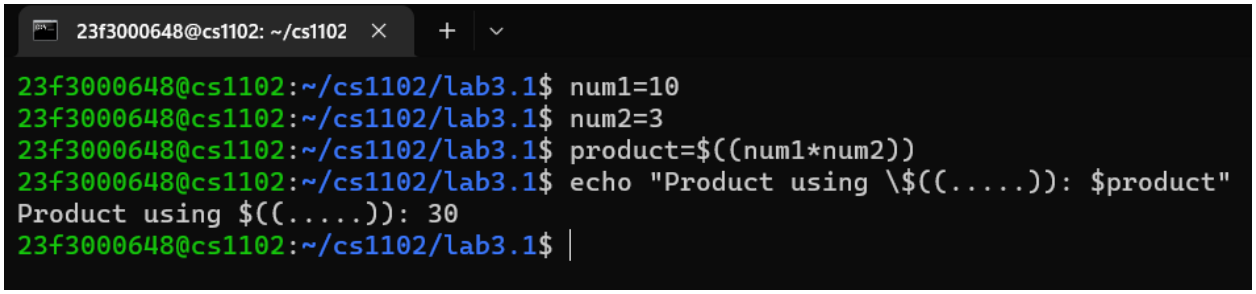
Explore four different ways Bash can perform basic math operations, including using the bc command.

Try adding two floating-point numbers with Bash.

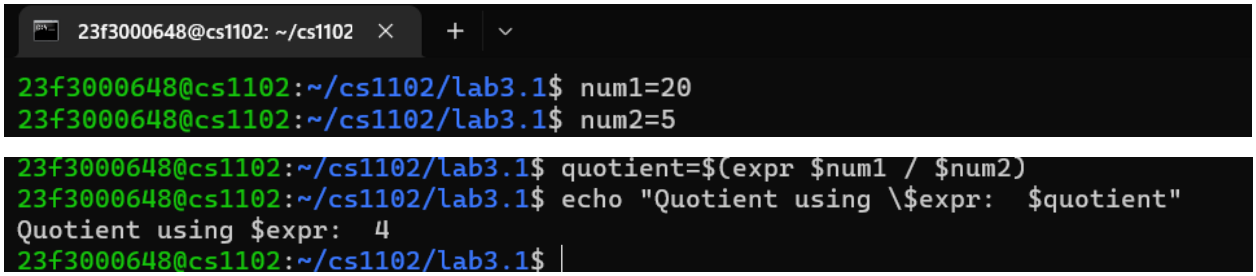
1. Using let for Integer Arithmetic



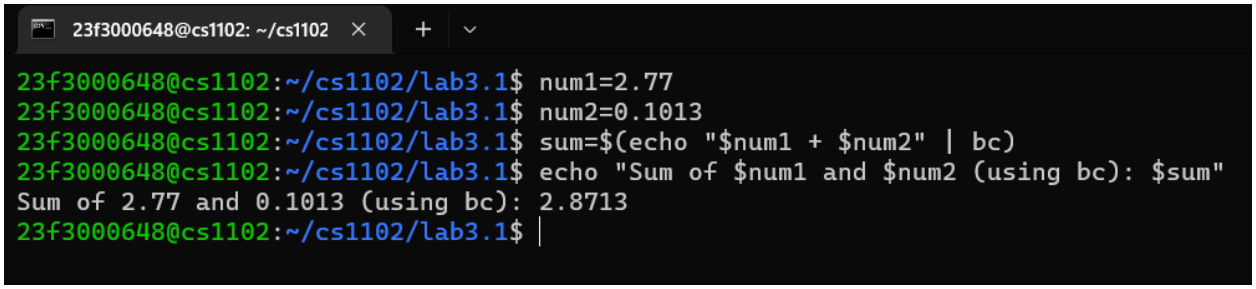
2. Using $((...)) for Integer Arithmetic



3. Using expr for Basic Integer Operations

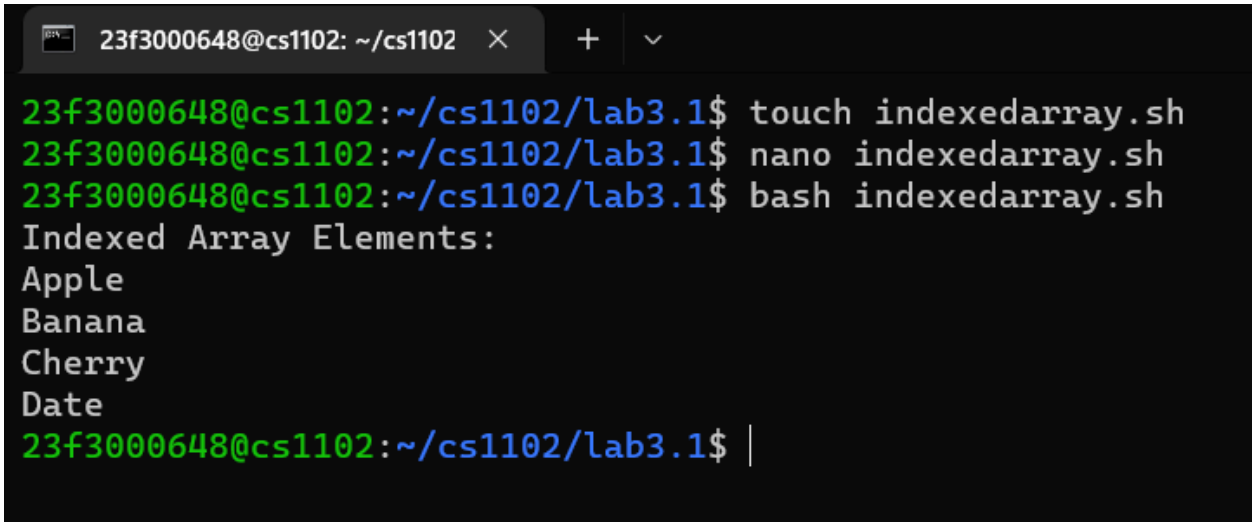


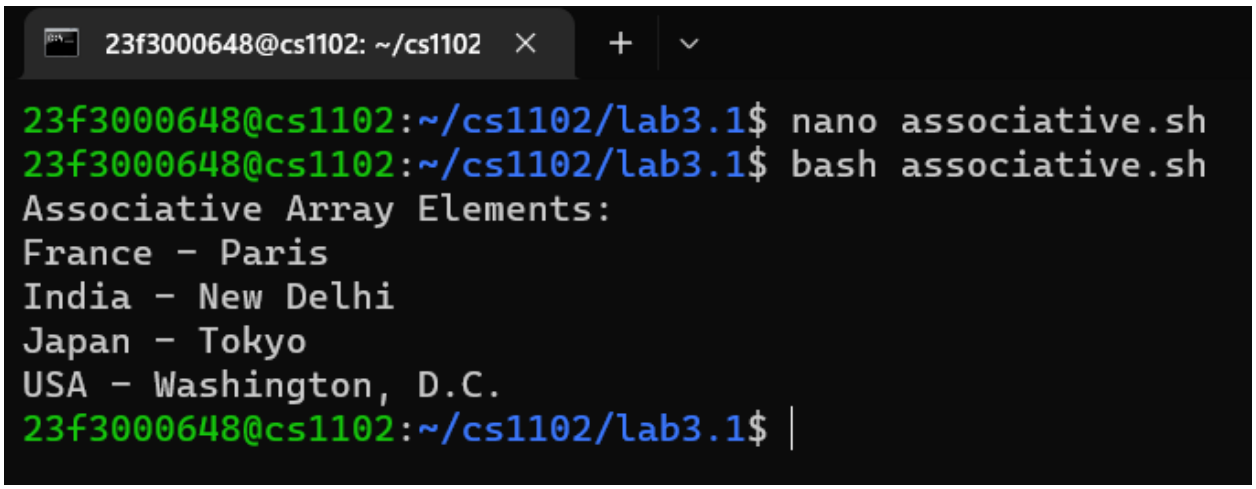
4. Using bc for Floating-Point Arithmetic

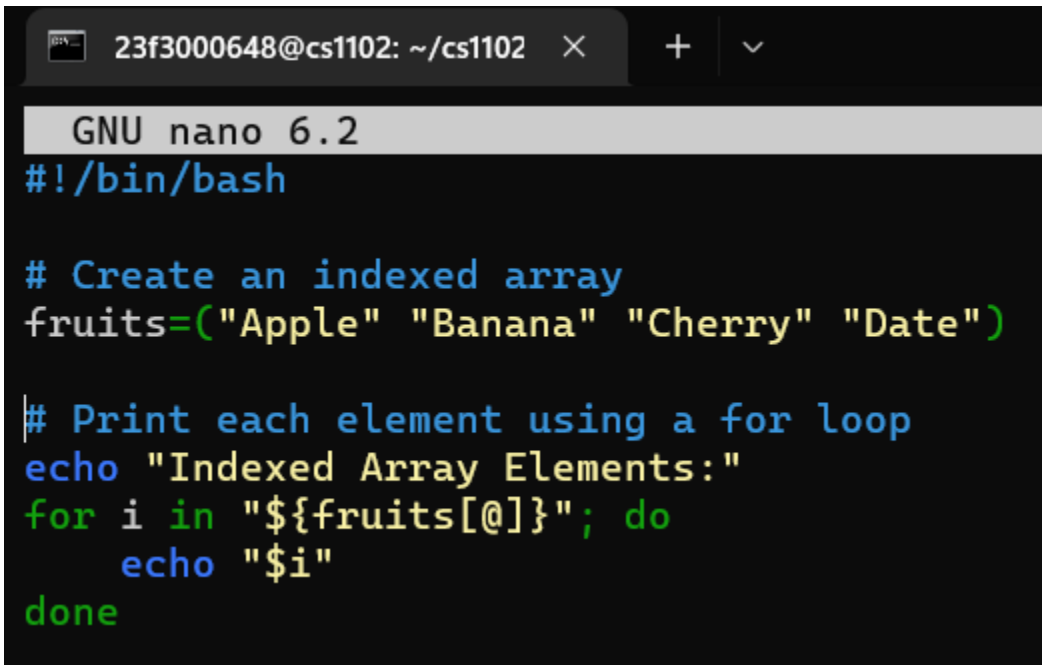


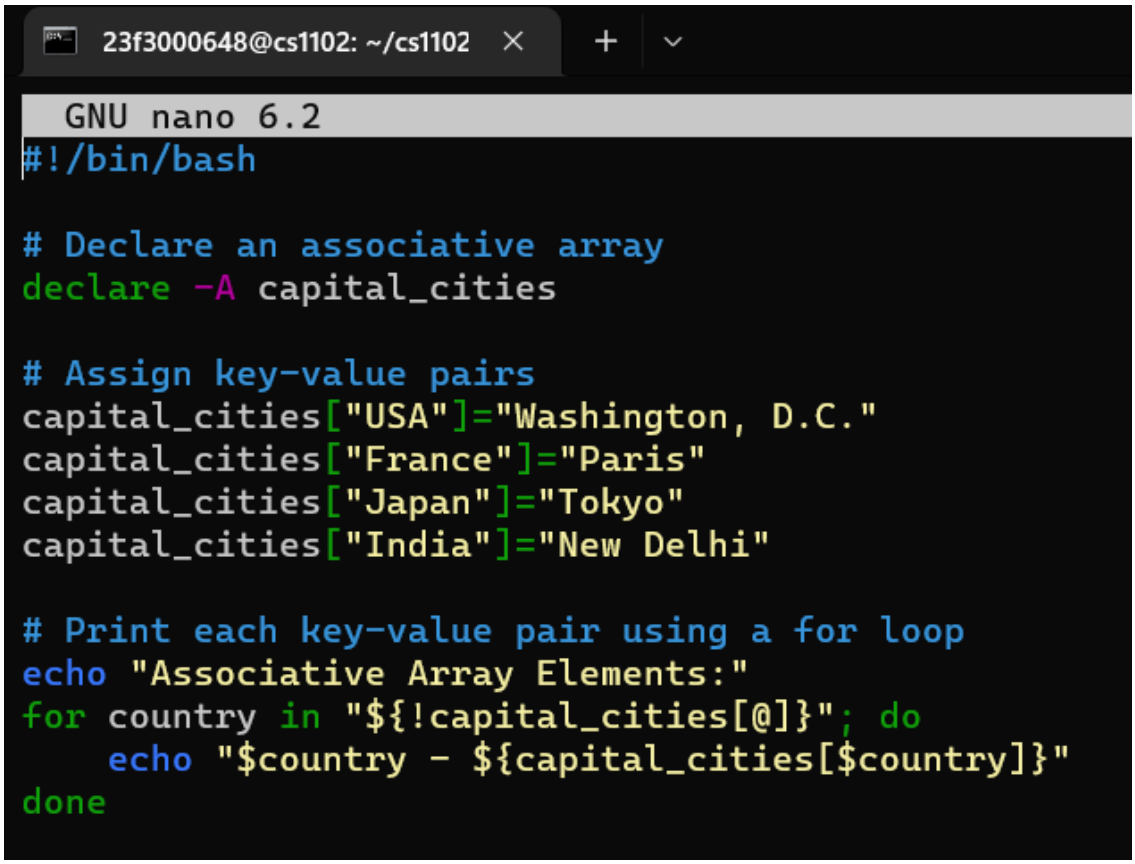
3.1.3 Arrays in Bash

Create indexed and associative arrays in Bash.
Demonstrate how to print elements of the arrays using a for loop.

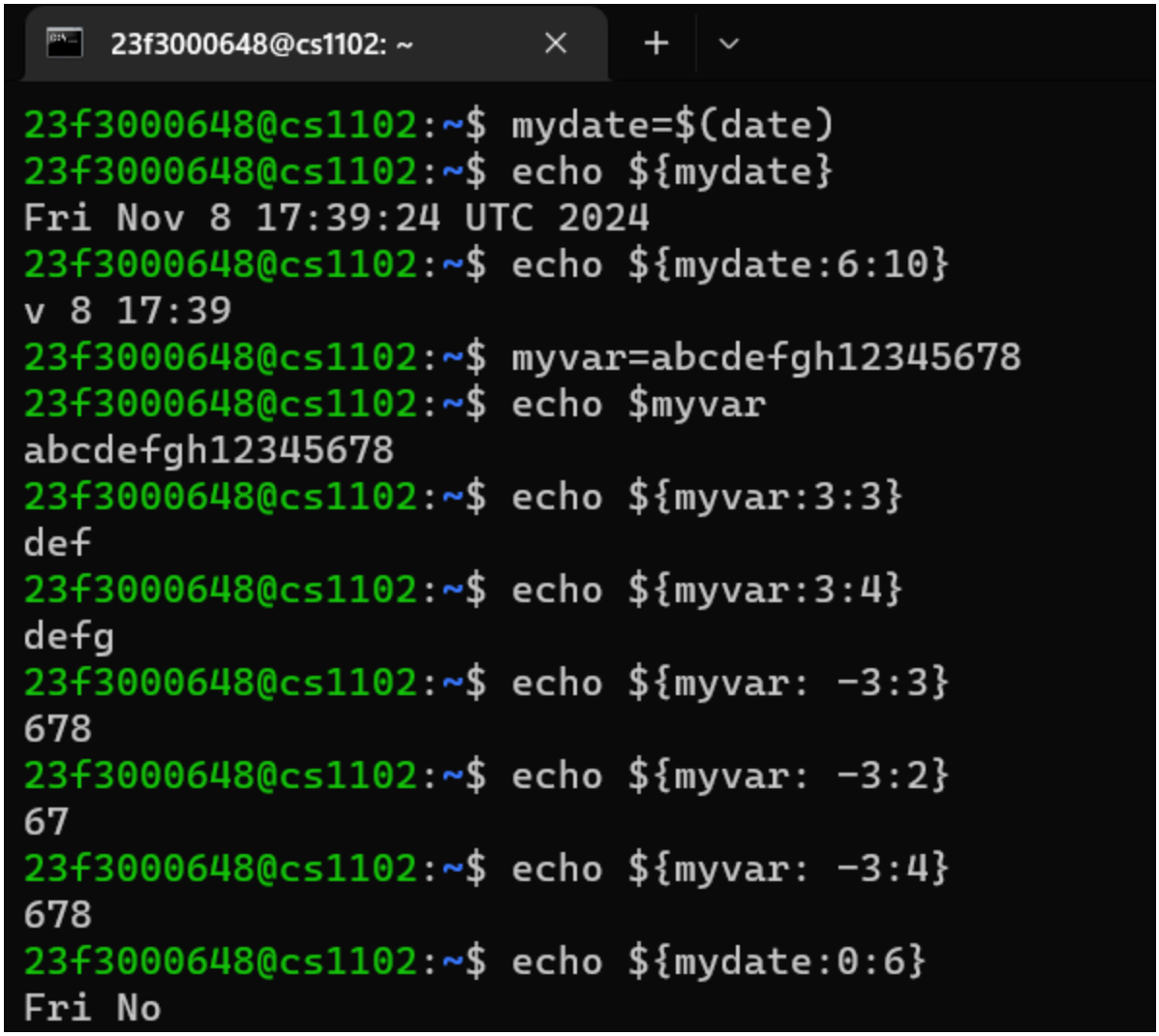

23f3000648@cs1102: ~/cs1102
23f3000648@cs1102:~/cs1102/lab3.1$ touch indexedarray.sh
23f3000648@cs1102:~/cs1102/lab3.1$ nano indexedarray.sh
23f3000648@cs1102:~/cs1102/lab3.1$ bash indexedarray.sh
Indexed Array Elements:
Apple
Banana
Cherry
Date
23f3000648@cs1102:~/cs1102/lab3.1$

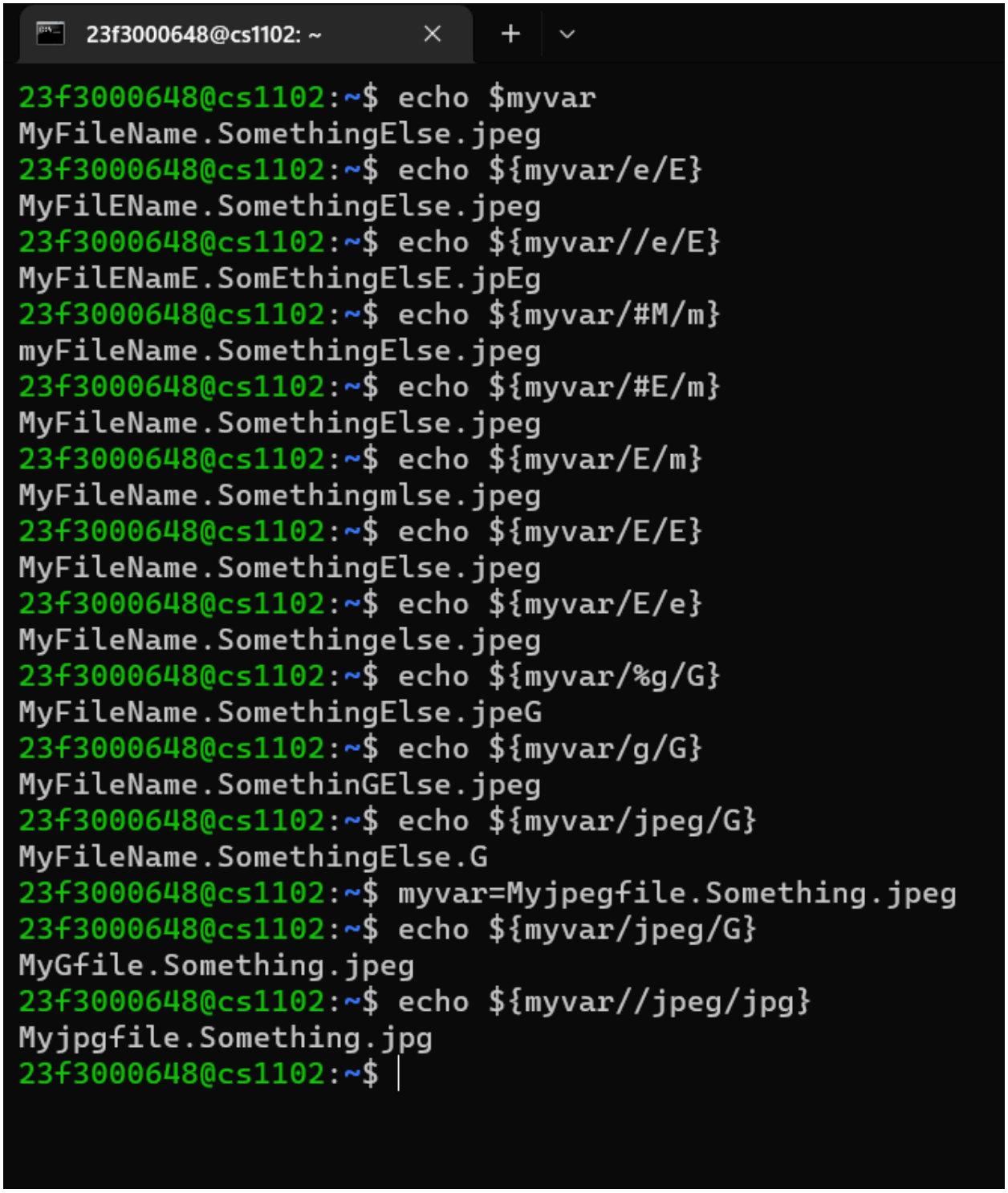

23f3000648@cs1102: ~/cs1102
23f3000648@cs1102:~/cs1102/lab3.1$ nano associative.sh
23f3000648@cs1102:~/cs1102/lab3.1$ bash associative.sh
Associative Array Elements:
France - Paris
India - New Delhi
Japan - Tokyo
USA - Washington, D.C.
23f3000648@cs1102:~/cs1102/lab3.1$

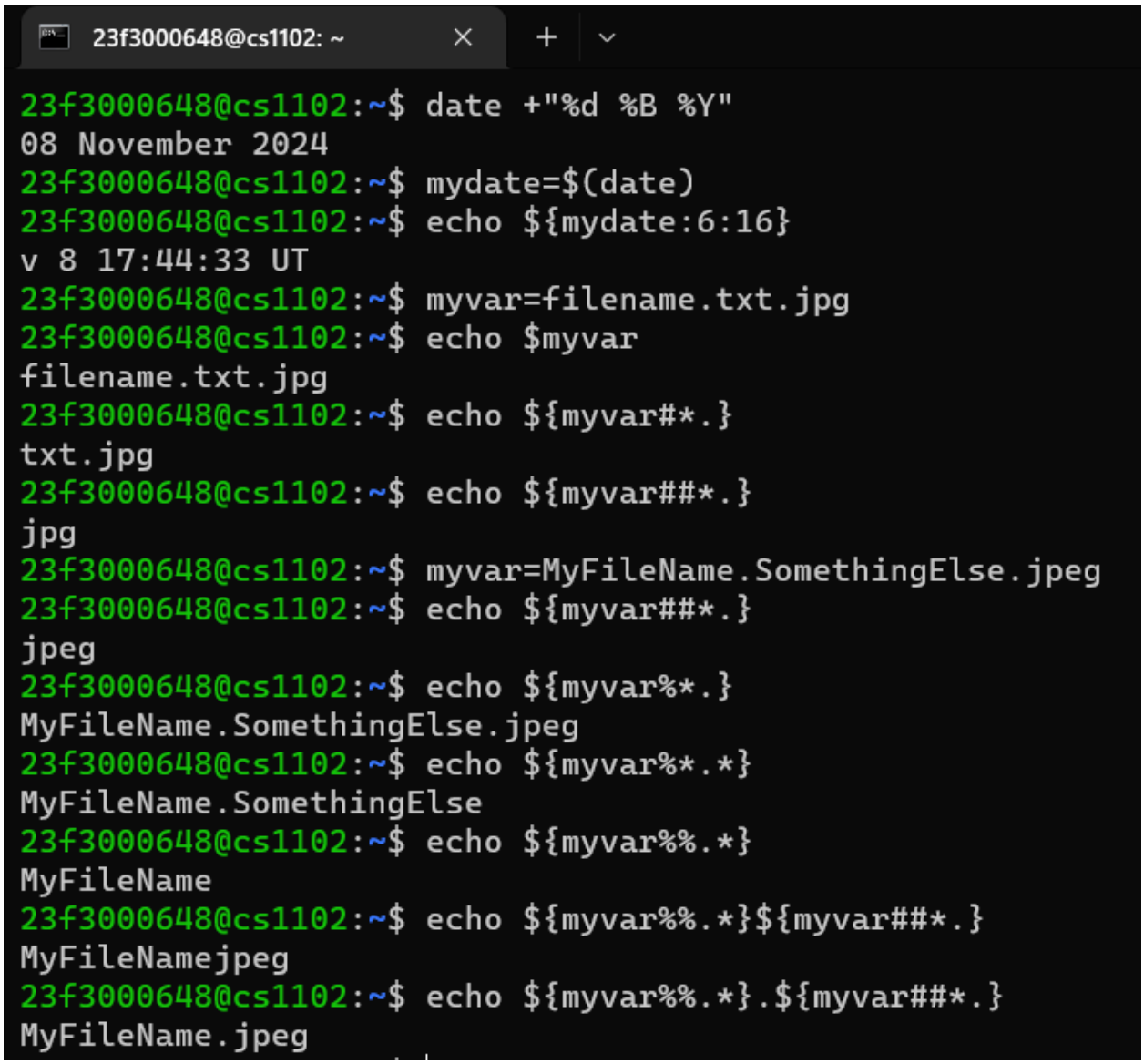

23f3000648@cs1102: ~/cs1102
GNU nano 6.2
#!/bin/bash
# Create an indexed array
fruits=("Apple" "Banana" "Cherry" "Date")
# Print each element using a for loop
echo "Indexed Array Elements:"
for i in "${fruits[@]}"; do
echo "$i"
done



3.1.4 Variable Manipulation in Bash


23f3000648@cs1102: ~
23f3000648@cs1102:~$ mydate=$(date)
23f3000648@cs1102:~$ echo ${mydate}
Fri Nov 8 17:39:24 UTC 2024
23f3000648@cs1102:~$ echo ${mydate:6:10}
v 8 17:39
23f3000648@cs1102:~$ myvar=abcdefgh12345678
23f3000648@cs1102:~$ echo $myvar
abcdefgh12345678
23f3000648@cs1102:~$ echo ${myvar:3:3}
def
23f3000648@cs1102:~$ echo ${myvar:3:4}
defg
23f3000648@cs1102:~$ echo ${myvar: -3:3}
678
23f3000648@cs1102:~$ echo ${myvar: -3:2}
67
23f3000648@cs1102:~$ echo ${myvar: -3:4}
678
23f3000648@cs1102:~$ echo ${mydate:0:6}
Fri No

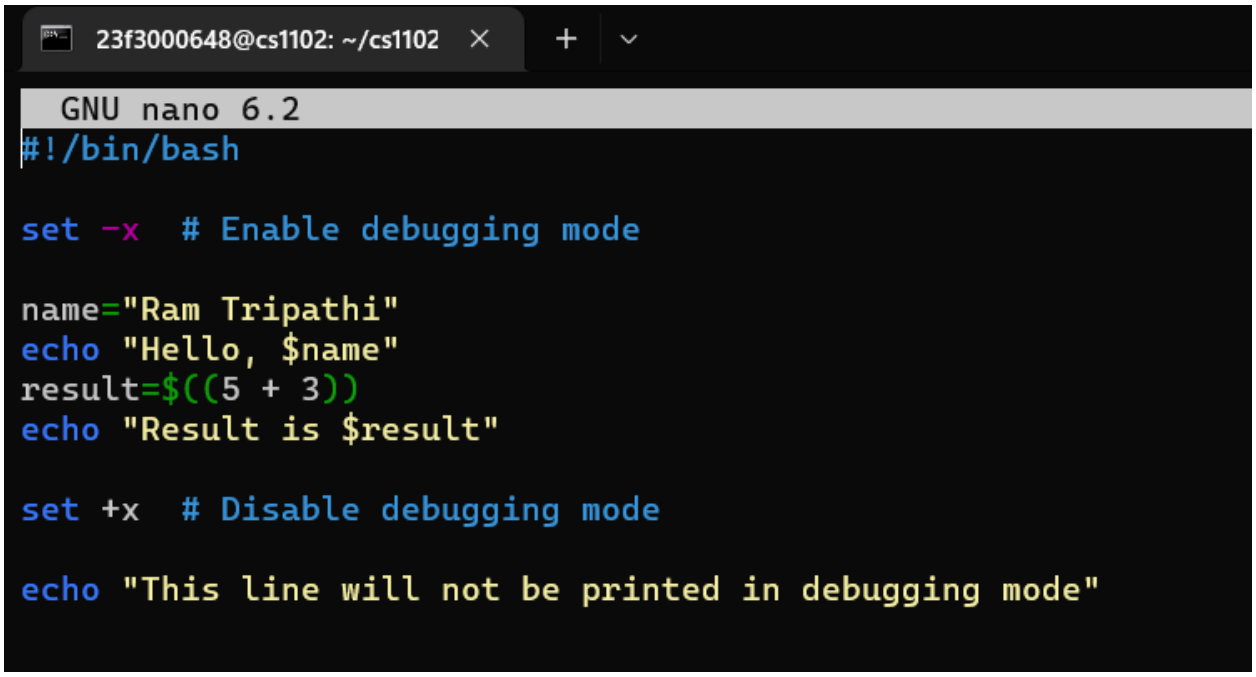

23f3000648@cs1102: ~
23f3000648@cs1102:~$ echo $myvar
MyFileName.SomethingElse.jpeg
23f3000648@cs1102:~$ echo ${myvar/e/E}
MyFilEName.SomethingElse.jpeg
23f3000648@cs1102:~$ echo ${myvar//e/E}
MyFilENamE.SomEthingElsE.jpEg
23f3000648@cs1102:~$ echo ${myvar/#M/m}
myFileName.SomethingElse.jpeg
23f3000648@cs1102:~$ echo ${myvar/#E/m}
MyFileName.SomethingElse.jpeg
23f3000648@cs1102:~$ echo ${myvar/E/m}
MyFileName.Somethingmlse.jpeg
23f3000648@cs1102:~$ echo ${myvar/E/E}
MyFileName.SomethingElse.jpeg
23f3000648@cs1102:~$ echo ${myvar/E/e}
MyFileName.Somethingelse.jpeg
23f3000648@cs1102:~$ echo ${myvar/%g/G}
MyFileName.SomethingElse.jpeG
23f3000648@cs1102:~$ echo ${myvar/g/G}
MyFileName.SomethinGElse.jpeg
23f3000648@cs1102:~$ echo ${myvar/jpeg/G}
MyFileName.SomethingElse.G
23f3000648@cs1102:~$ myvar=Myjpegfile.Something.jpeg
23f3000648@cs1102:~$ echo ${myvar/jpeg/G}
MyGfile.Something.jpeg
23f3000648@cs1102:~$ echo ${myvar//jpeg/jpg}
Myjpgfile.Something.jpg
23f3000648@cs1102:~$

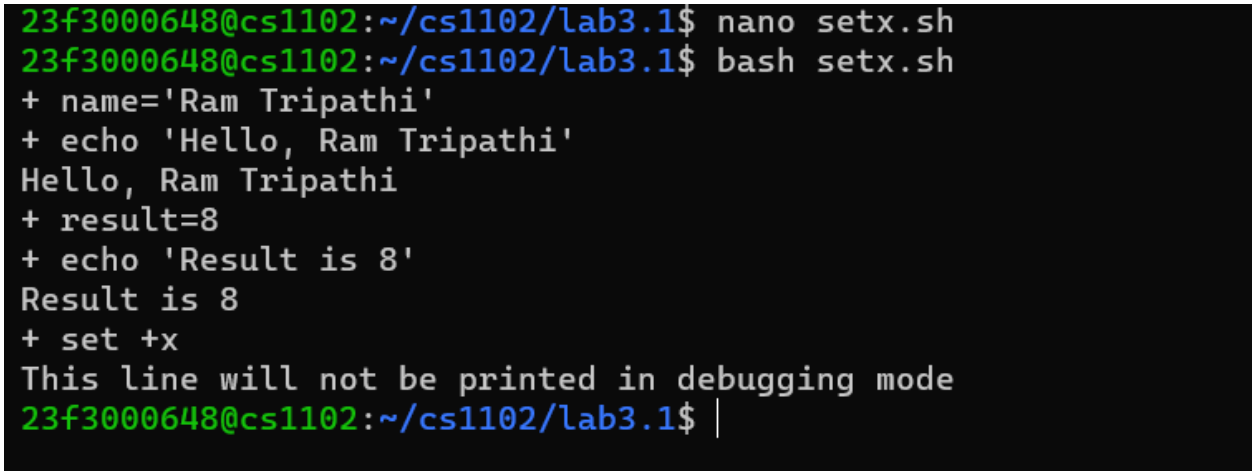


3.1.5 set -x

Explain what set -x does.
The set command discussed in shell scripts enables debug mode. When you include x in your script. It activates the trace feature, which causes the shell to print each command before it is executed

*When you add set -x at the beginning of your script or within a specific section. It turns on debugging mode for that part of the script.
*Once debugging mode is enabled, the shell prints each command (alomg with its expanded arguments) to the standard error(usuallly the terminal) before executing it.





3.1.6 read Command Use Case
Report various use cases of the read command and assigning its value to a variable

1. Reading a Single Input from the User

```
23f3000648@cs1102: ~/cs1102
23f3000648@cs1102:~/cs1102/lab3.1$ read -p "Enter your name : " name
Enter your name : Ram Tripathi
23f3000648@cs1102:~/cs1102/lab3.1$ echo "Hello : $namw"
Hello :
23f3000648@cs1102:~/cs1102/lab3.1$ echo "Hello : $name"
Hello : Ram Tripathi
23f3000648@cs1102:~/cs1102/lab3.1$
```

2. Reading Multiple Values into Multiple Variables

```
23f3000648@cs1102: ~/cs1102
23f3000648@cs1102:~/cs1102/lab3.1$ read -p "Enter your first name and last name :" first_name last_name
Enter your first name and last name :Ram Tripathi
23f3000648@cs1102:~/cs1102/lab3.1$ echo "Hello, $first_name $last_name !"
Hello, Ram Tripathi !
23f3000648@cs1102:~/cs1102/lab3.1$
```

3. Reading with a Timeout

```
23f3000648@cs1102: ~/cs1102
23f3000648@cs1102:~/cs1102/lab3.1$ read -t 5 -p "Enter your favorite color: " color
Enter your favorite color: red
23f3000648@cs1102:~/cs1102/lab3.1$ echo "Your Favorite color is: $(color:-'not provided')."
color:-not provided: command not found
Your Favorite color is: .
23f3000648@cs1102:~/cs1102/lab3.1$ echo "Your Favorite color is: ${color:-'not provided'}."
Your Favorite color is: red.
23f3000648@cs1102:~/cs1102/lab3.1$
```

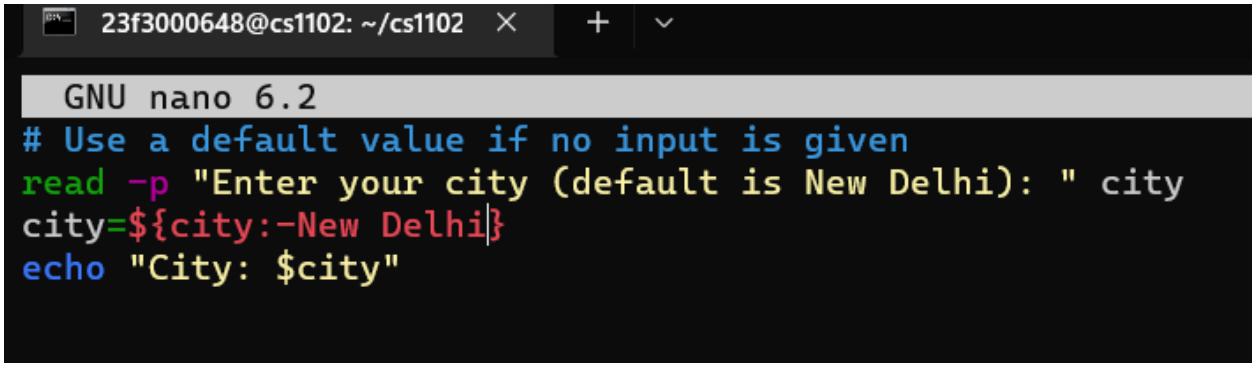
4 . Silent Input (Useful for Passwords)

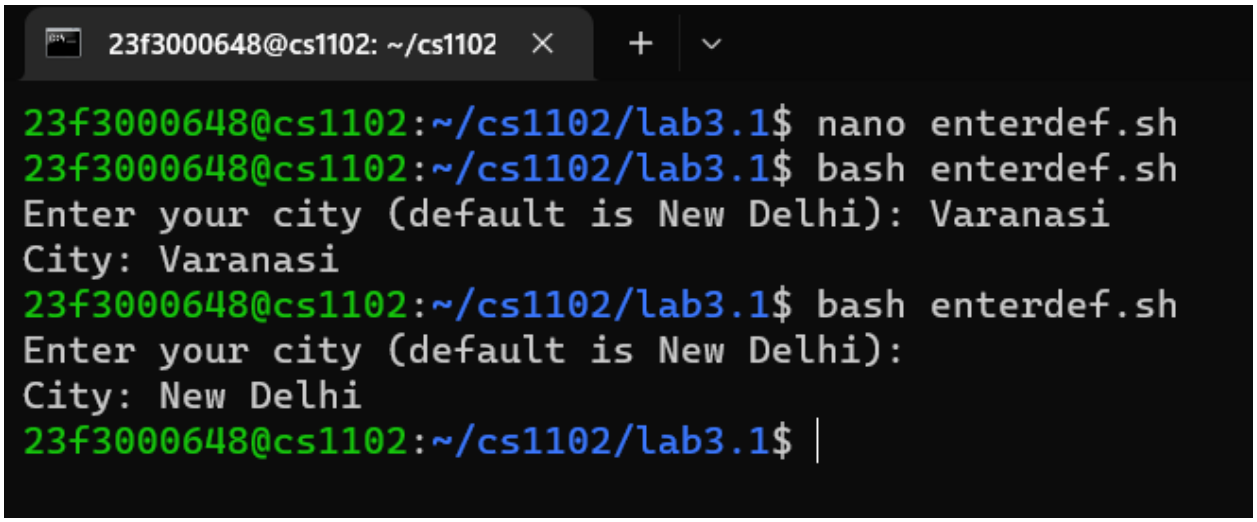
```
23f3000648@cs1102:~/cs1102/lab3.1$ read -s -p "Your password: " password
Your password: 23f3000648@cs1102:~/cs1102/lab3.1$
```

5. Reading a Limited Number of Characters

```
23f3000648@cs1102: ~/cs1102
23f3000648@cs1102:~/cs1102/lab3.1$ read -n 4 -p "Enter your code recieved: " code
Enter your code recieved: 123423f3000648@cs1102:~/cs1102/lab3.1$
23f3000648@cs1102:~/cs1102/lab3.1$ echo -e "\n You entered: $code"

 You entered: 1234
23f3000648@cs1102:~/cs1102/lab3.1$
```

6. Reading Input with a Default Value Using Parameter Expansion



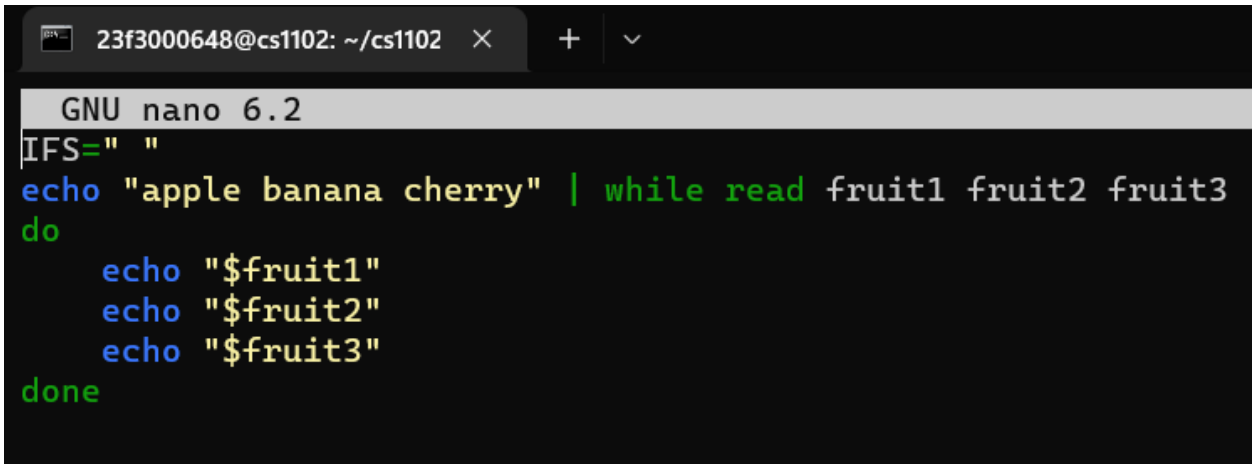


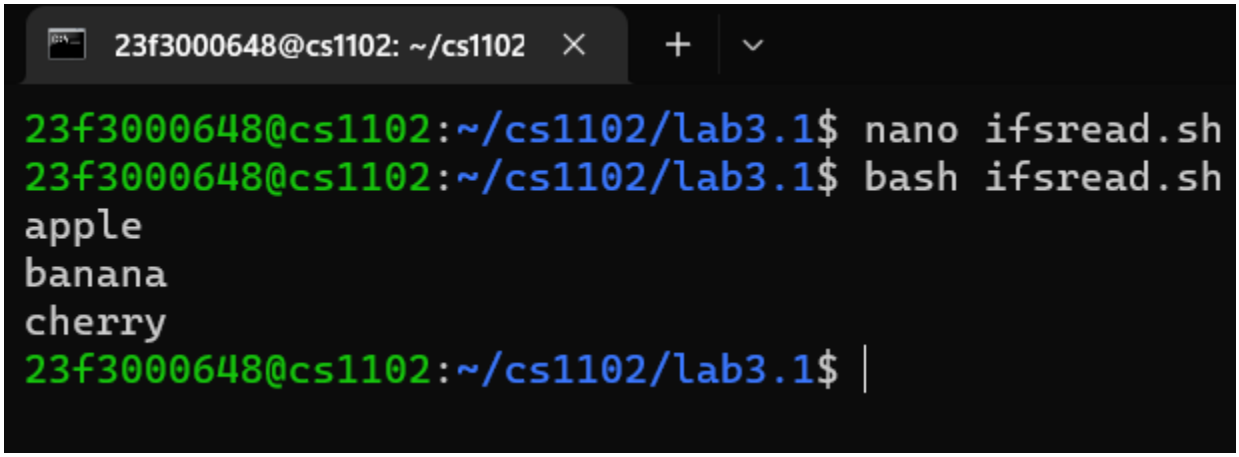
3.1.7 IFS and Its Use

Define IFS and provide at least two examples of its usage

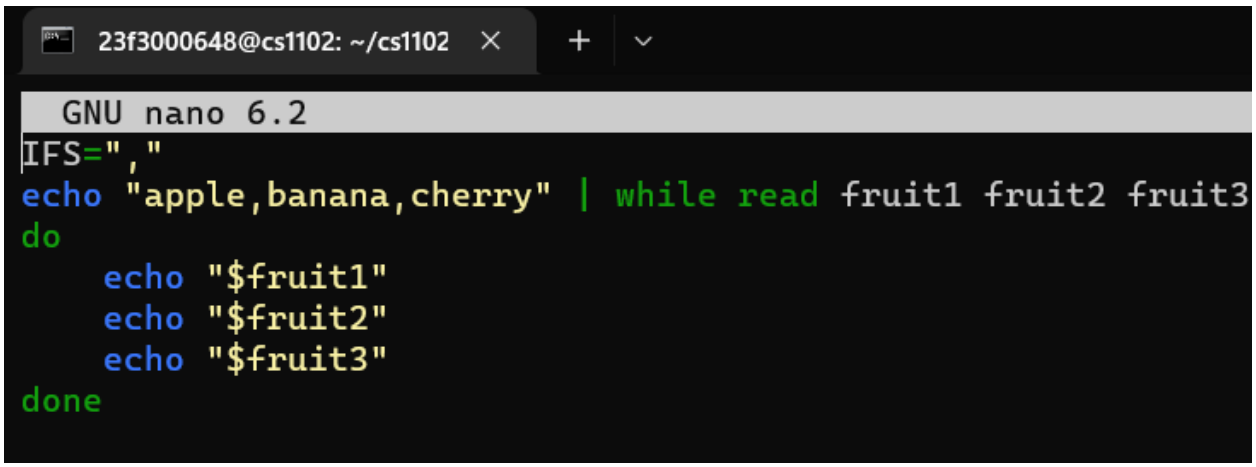
The IFS variable in Linux (specifically in Bash and other shell environments) controls the splitting behavior of input, particularly when using commands like read, for, and other loops. When you use these commands to handle text data, the IFS dictates how the data is divided into separate fields or tokens.

Using IFS with the `read` command (default delimiter space)





Using IFS to split by a custom delimiter



```
23f3000648@cs1102:~/cs1102/lab3.1$ nano ifsreadcustom.sh
23f3000648@cs1102:~/cs1102/lab3.1$ bash ifsreadcustom.sh
apple
banana
cherry
23f3000648@cs1102:~/cs1102/lab3.1$
```

3.1.8 Reading File Content

Explain how read, IFS, and while are used together to read the content of a file word by word. Show how field separators can be changed, possibly using a log file shared in a previous session.

Using IFS, a while loop, and the read command, one can read a file word by word in Linux by processing each line and dividing it into distinct words (or tokens) based on a predefined delimiter.

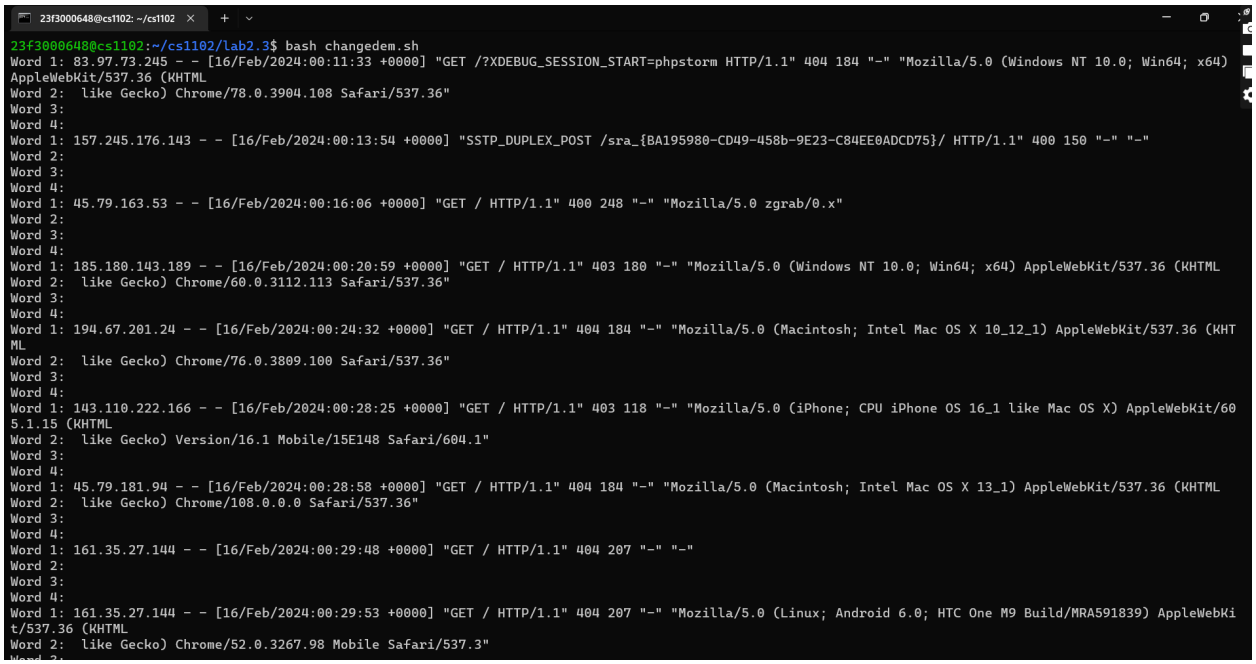
These elements cooperate as follows:

read: Takes one input line at a time. You can process every line of the file in a loop.
The internal field separator, or IFS, regulates how the line is divided into distinct words. It divides by spaces, tabs, and newlines by default, but you may change it to use other delimiters like semicolons or commas.
while loop: Continues processing until the end of the file is reached. Each iteration processes a single line from the file.

```
23f3000648@cs1102:~/cs1102/lab2.3$ ls
access.log  uiqueIP.txt
23f3000648@cs1102:~/cs1102/lab2.3$ nano changedem.sh
23f3000648@cs1102:~/cs1102/lab2.3$ nano changedem.sh
23f3000648@cs1102:~/cs1102/lab2.3$ bash changedem.sh
changedem.sh: line 13: logfile.txt: No such file or directory
23f3000648@cs1102:~/cs1102/lab2.3$ nano changedem.sh
23f3000648@cs1102:~/cs1102/lab2.3$ bash changedem.sh
```

Using access.log file of lab2

Using , as a delimiter

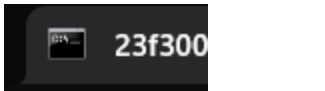


3.2 Loops

For Loops

Create a for loop to print numbers from 10 to 245.

```
23f3000648@cs1102:~/cs1102$ mkdir lab3.2
23f3000648@cs1102:~/cs1102$ ls
3.2  lab  lab1  lab1.2  lab1.3  lab2.1  lab2.2  lab2.3  lab2.4  lab3.1  lab3.2  lab3.4
23f3000648@cs1102:~/cs1102$ cd lab3.2
23f3000648@cs1102:~/cs1102/lab3.2$ ls
23f3000648@cs1102:~/cs1102/lab3.2$ for ((i=0; i<=245 ; i++))
> do
> echo $i
> done
```




```
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
```

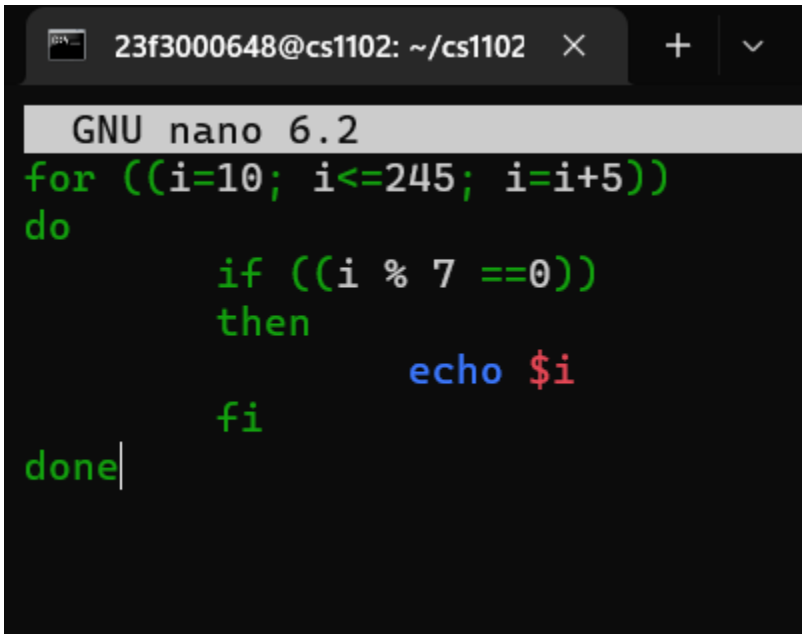
```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
```

Modify the for loop to print numbers from 10 to 245 with a step of 5

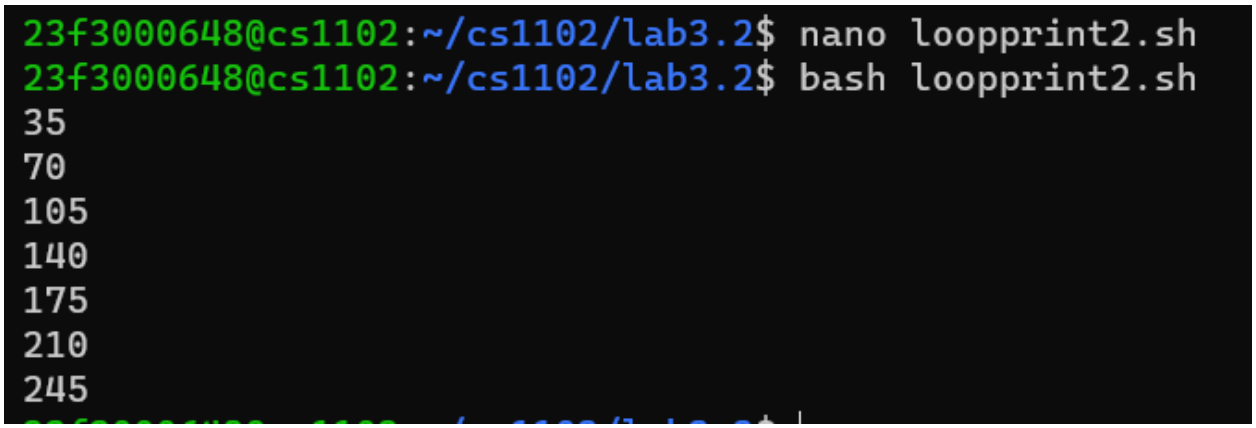
```
23f3000648@cs1102:~/cs1102/lab3.2$ nano loopprint2.sh
23f3000648@cs1102:~/cs1102/lab3.2$ bash loopprint2.sh
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
110
115
120
125
130
135
140
145
150
155
160
```

```
75
80
85
90
95
100
105
110
115
120
125
130
135
140
145
150
155
160
165
170
175
180
185
190
195
200
205
210
215
220
225
230
235
240
245
```

Modify the for loop to print numbers from 10 to 245 with a step of 5

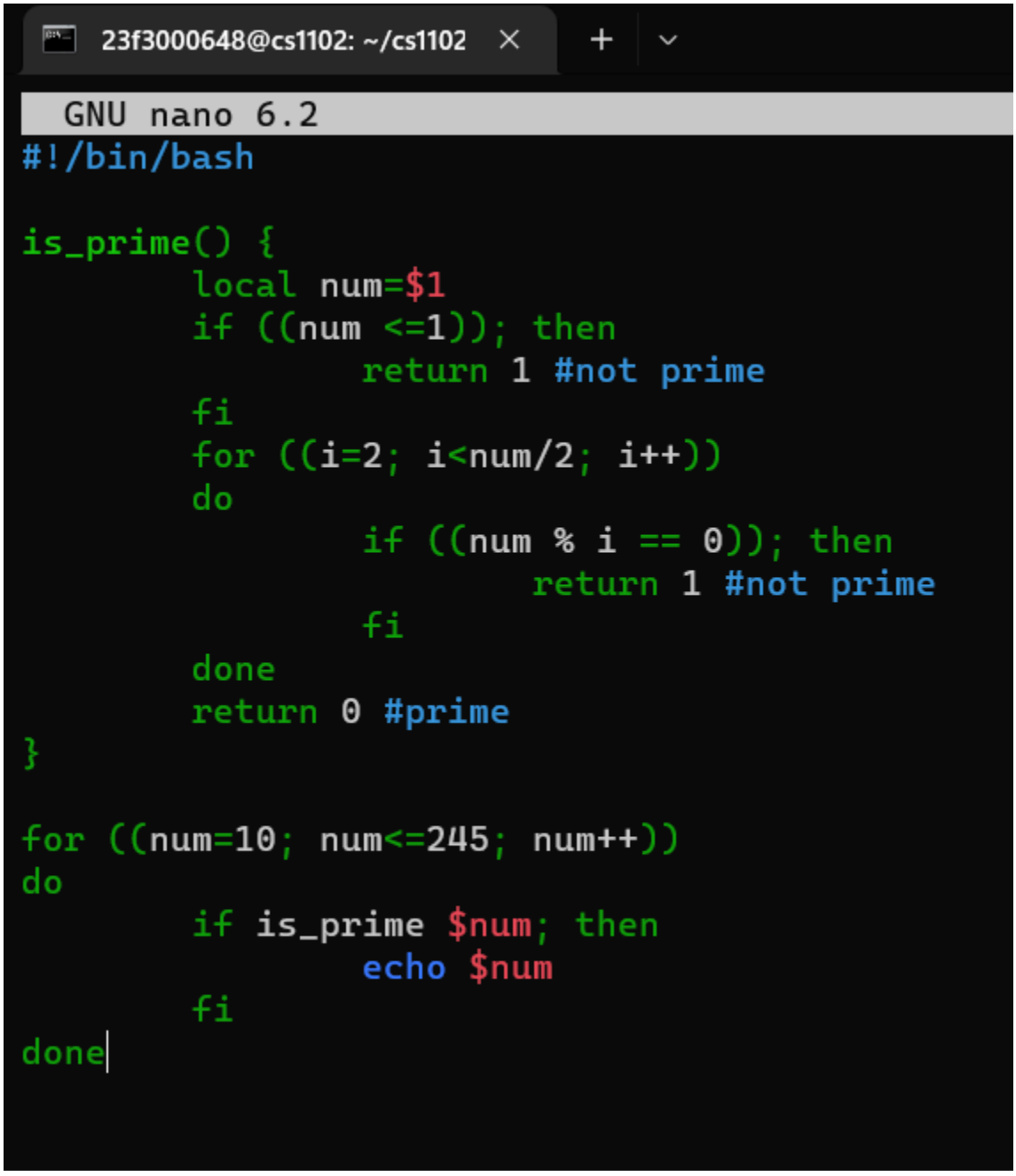


Further modify the for loop to print numbers from 10 to 245 if divisible by 5 and 7.



3.2.2 Prime Numbers

Add an if condition to the for loop to print only prime numbers in the range 10 to 245.

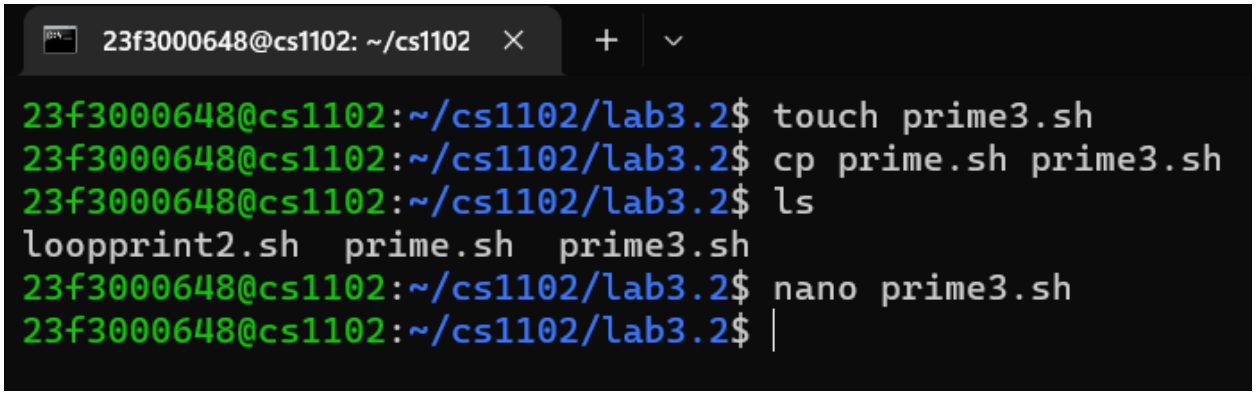


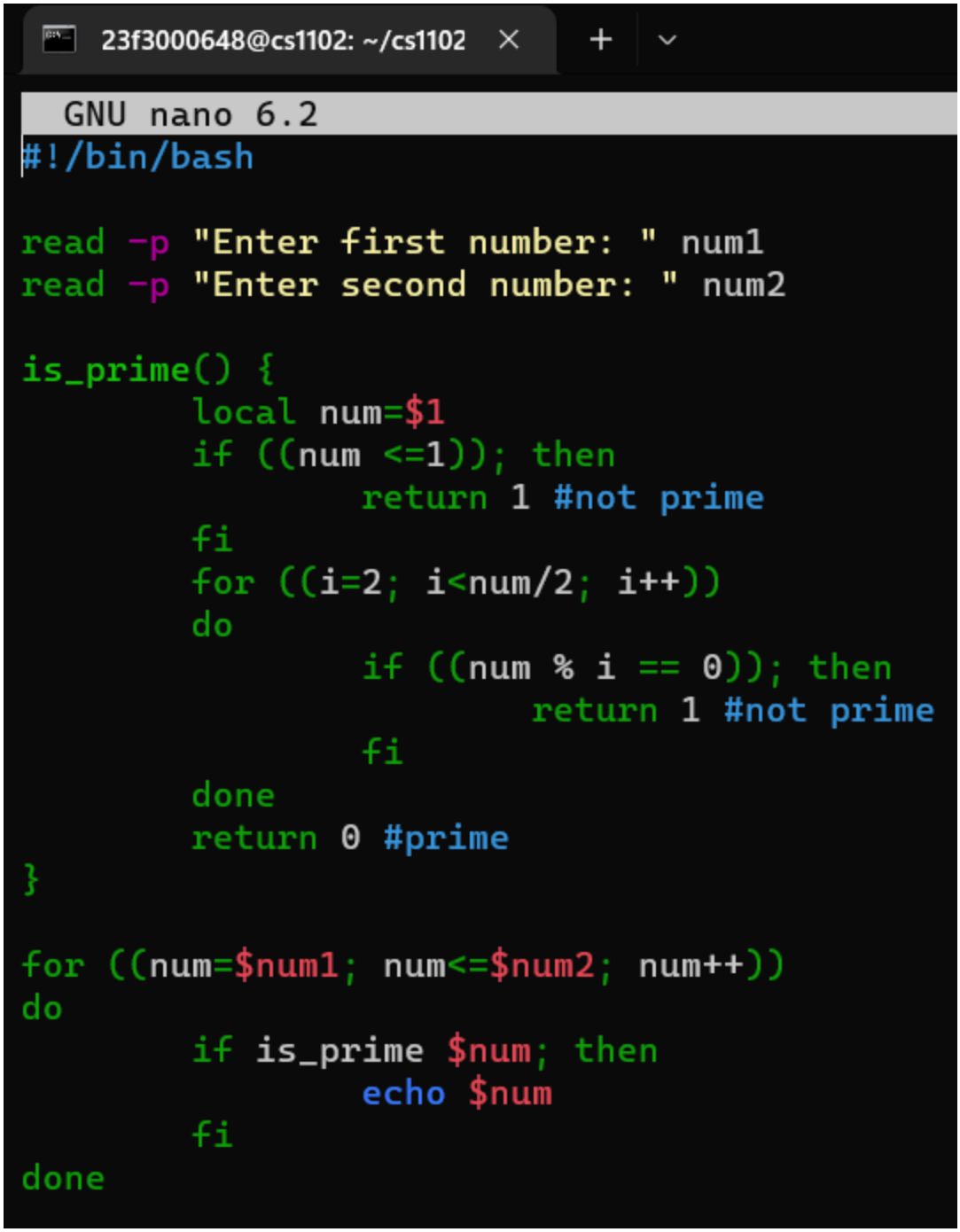
```
23f3000648@cs1102:~/cs1102/lab3.2$ nano prime.sh
23f3000648@cs1102:~/cs1102/lab3.2$ bash prime.sh
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
83
89
97
101
103
107
109
113
127
131
137
139
149
151
157
163
167
173
179
```

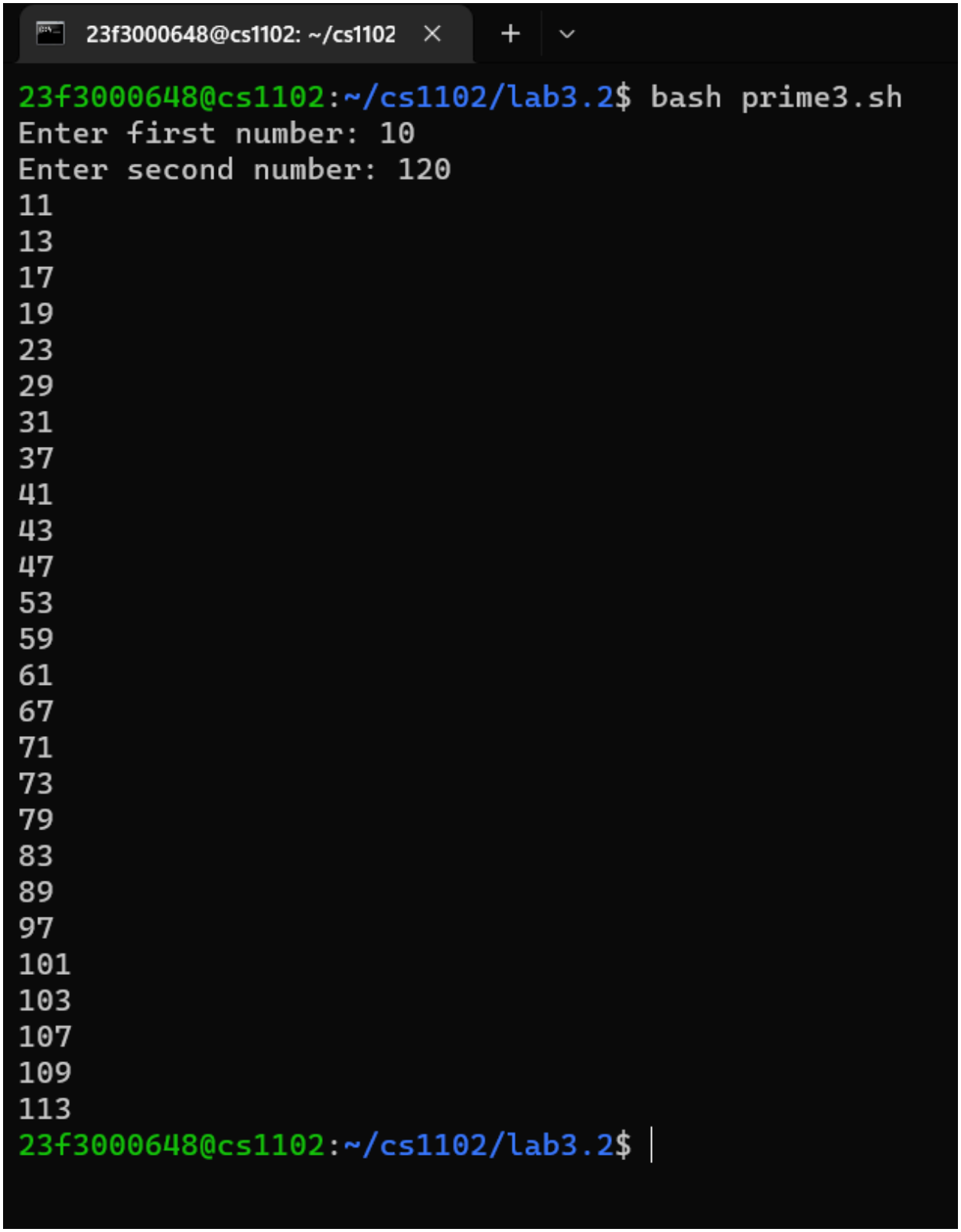
```
157
163
167
173
179
181
191
193
197
199
211
223
227
229
233
239
241
```

3.2.3 Odd Numbers Script

Create a script that asks for a range (starting point and end point) and prints all odd numbers within that range.

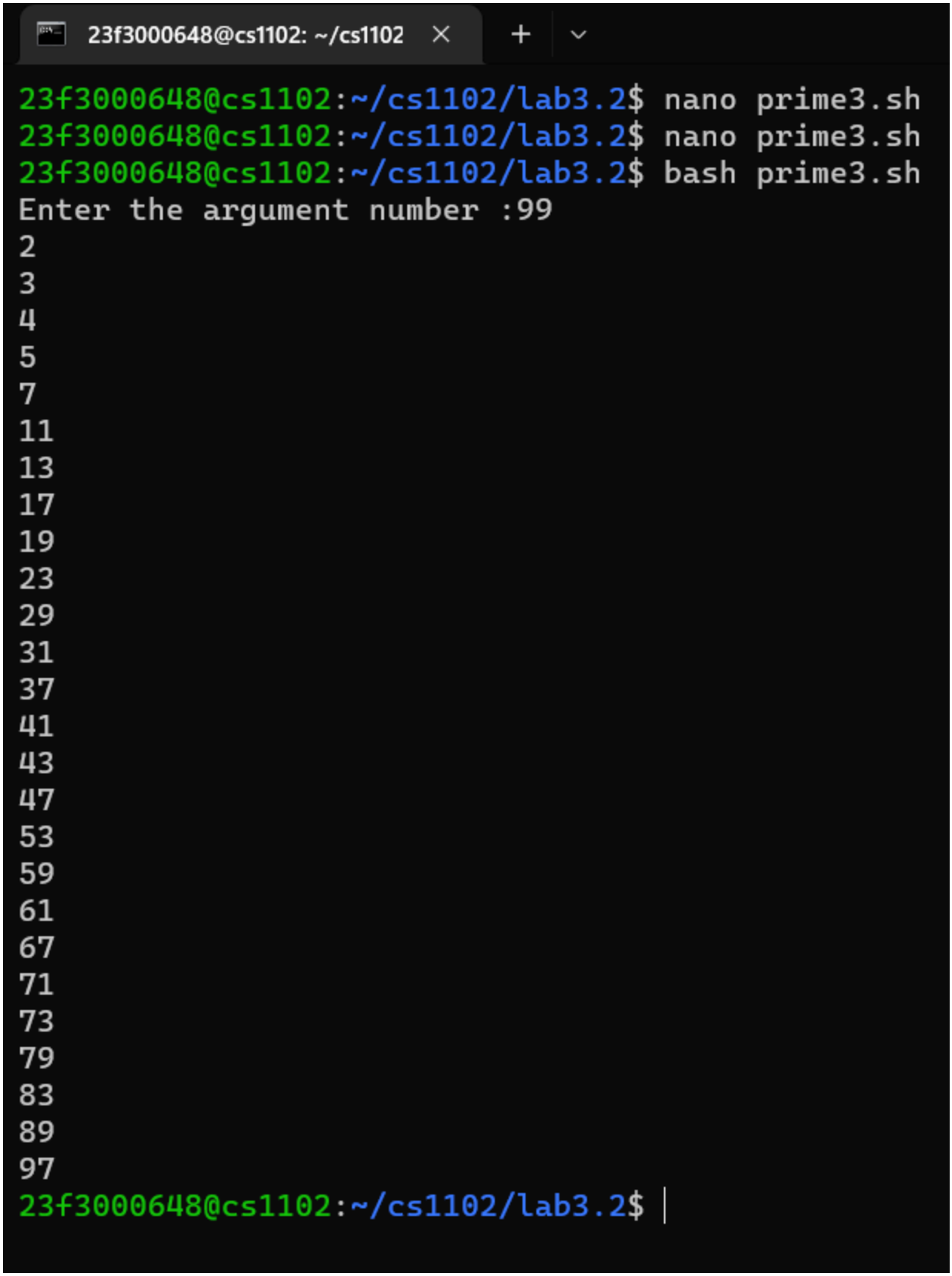


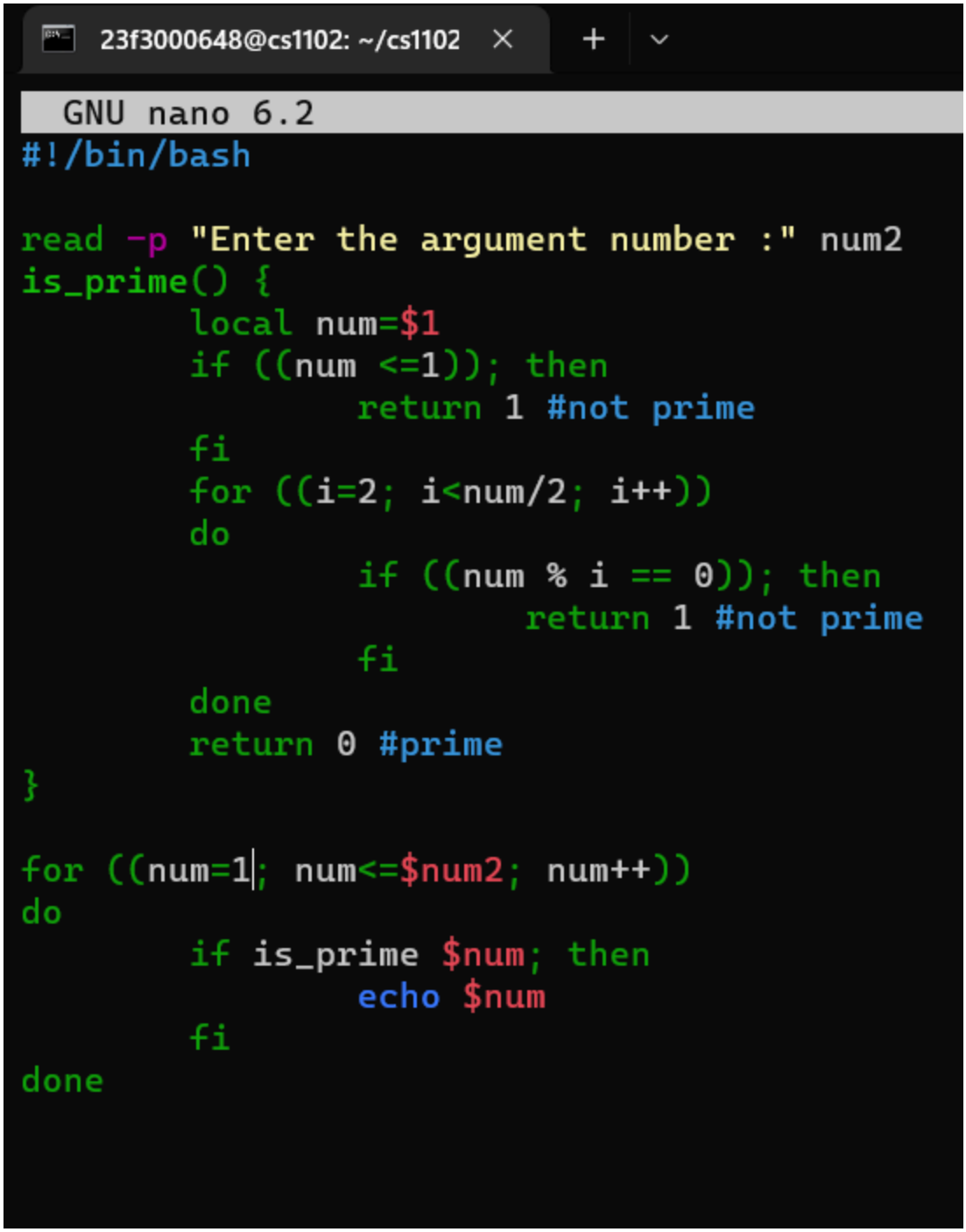

23f3000648@cs1102: ~/cs1102
GNU nano 6.2
#!/bin/bash
read -p "Enter first number: " num1
read -p "Enter second number: " num2
is_prime() {
local num=$1
if ((num <=1)); then
return 1 #not prime
fi
for ((i=2; i<num/2; i++))
do
if ((num % i == 0)); then
return 1 #not prime
fi
done
return 0 #prime
}
for ((num=$num1; num<=$num2; num++))
do
if is_prime $num; then
echo $num
fi
done



3.2.4 Prime Numbers Script

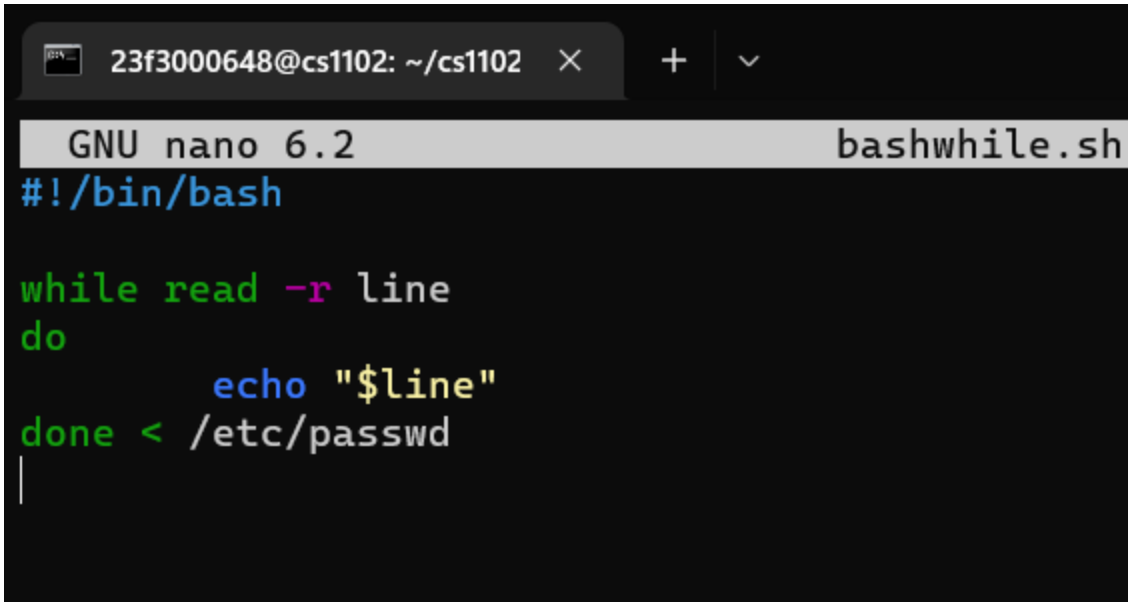
Create a script that takes an argument and prints all prime numbers between 1 and the argument number.

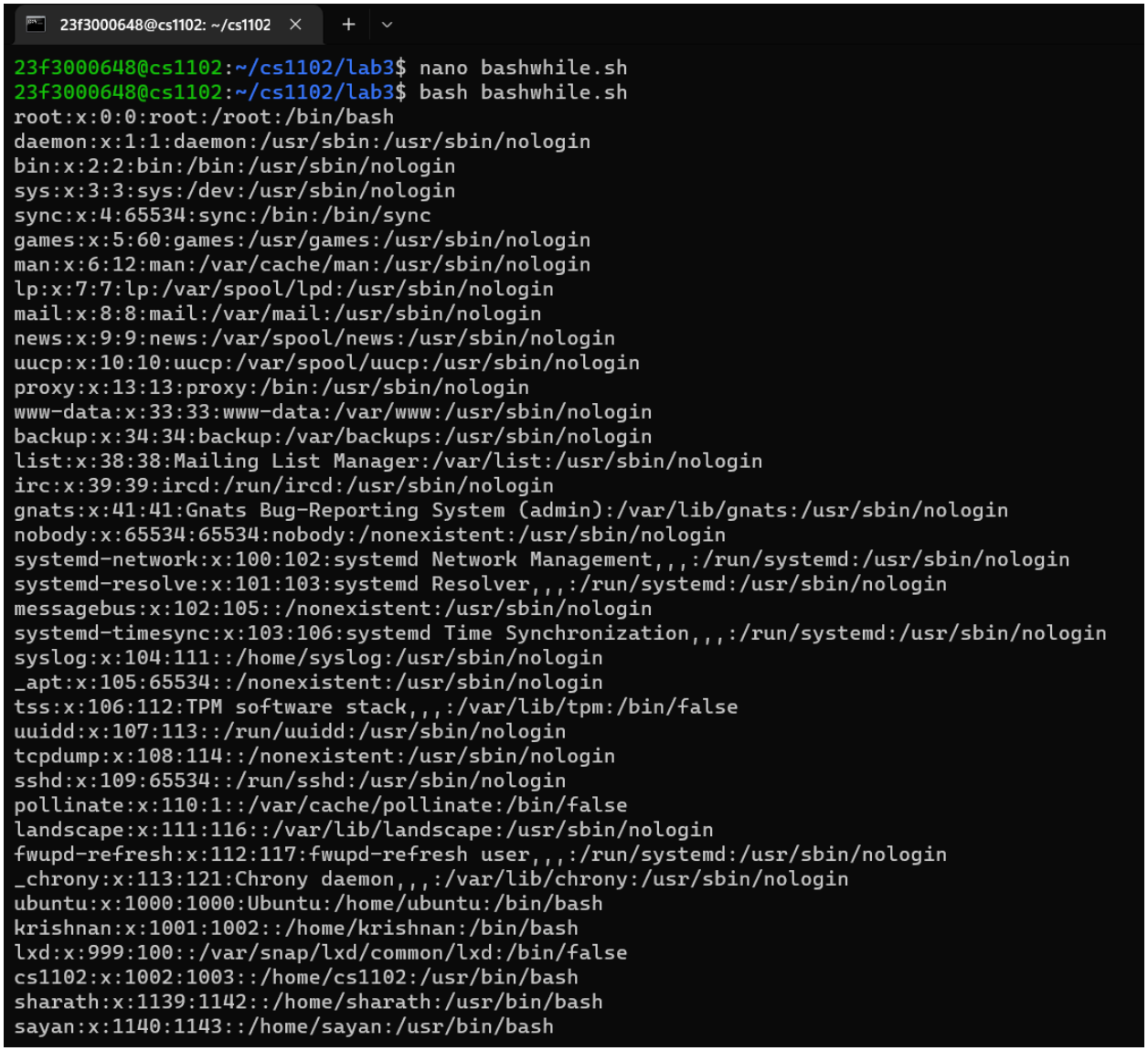

23f3000648@cs1102: ~/cs1102
23f3000648@cs1102:~/cs1102/lab3.2$ nano prime3.sh
23f3000648@cs1102:~/cs1102/lab3.2$ nano prime3.sh
23f3000648@cs1102:~/cs1102/lab3.2$ bash prime3.sh
Enter the argument number :99
2
3
4
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
83
89
97
23f3000648@cs1102:~/cs1102/lab3.2$



3.3 Reading Files

Reading /etc/passwd File
Create a while loop to read the /etc/passwd text file


23f3000648@cs1102: ~/cs1102
GNU nano 6.2 bashwhile.sh
#!/bin/bash
while read -r line
do
echo "$line"
done < /etc/passwd



Using IFS to Read /etc/passwd File

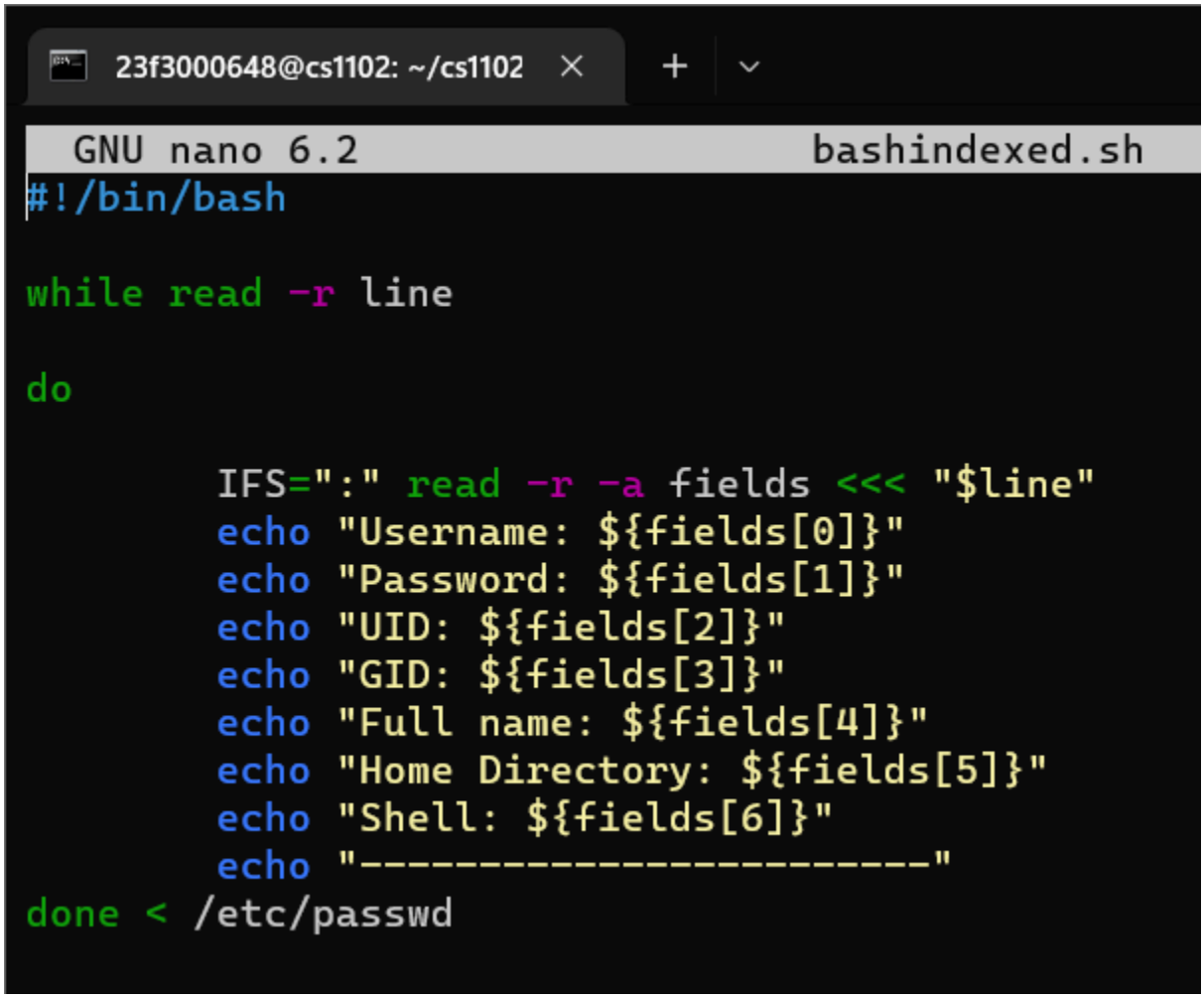
Create a while loop using IFS to use ':' as a separator to read /etc/passwd and print its contents word by word.


23f3000648@cs1102: ~/cs1102
GNU nano 6.2
bashreadscriptpasswd.sh
while IFS=: read -r username password uid gid fullname home shell
do
echo "Username: $username"
echo "Password: $password"
echo "UID: $uid"
echo "GID: $gid"
echo "Full Name : $fullname"
echo "Home Directory : $home"
echo "Shell: $shell"
echo "----------"
done < /etc/passwd

```
23f3000648@cs1102:~/cs1102/lab3$ nano bashreadscriptpasswd.sh
23f3000648@cs1102:~/cs1102/lab3$ bash bashreadscriptpasswd.sh
Username: root
Password: x
UID: 0
GID: 0
Full Name : root
Home Directory : /root
Shell: /bin/bash
-----------------------
Username: daemon
Password: x
UID: 1
GID: 1
Full Name : daemon
Home Directory : /usr/sbin
Shell: /usr/sbin/nologin
-----------------------
Username: bin
Password: x
UID: 2
GID: 2
Full Name : bin
Home Directory : /bin
Shell: /usr/sbin/nologin
-----------------------
Username: sys
Password: x
UID: 3
GID: 3
Full Name : sys
Home Directory : /dev
Shell: /usr/sbin/nologin
-----------------------
Username: sync
Password: x
UID: 4
GID: 65534
Full Name : sync
```

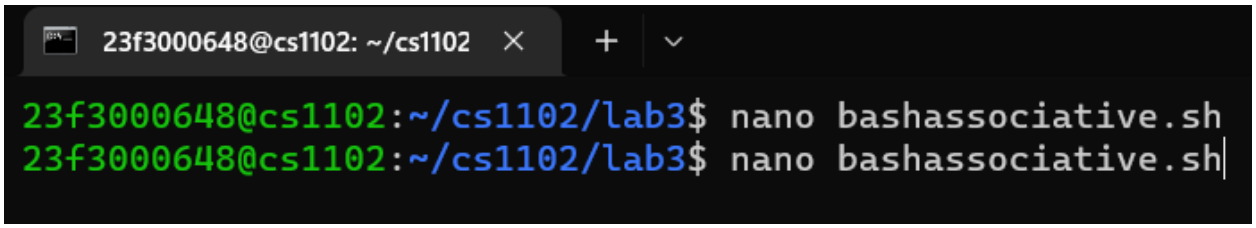
Using Associative and Indexed Arrays

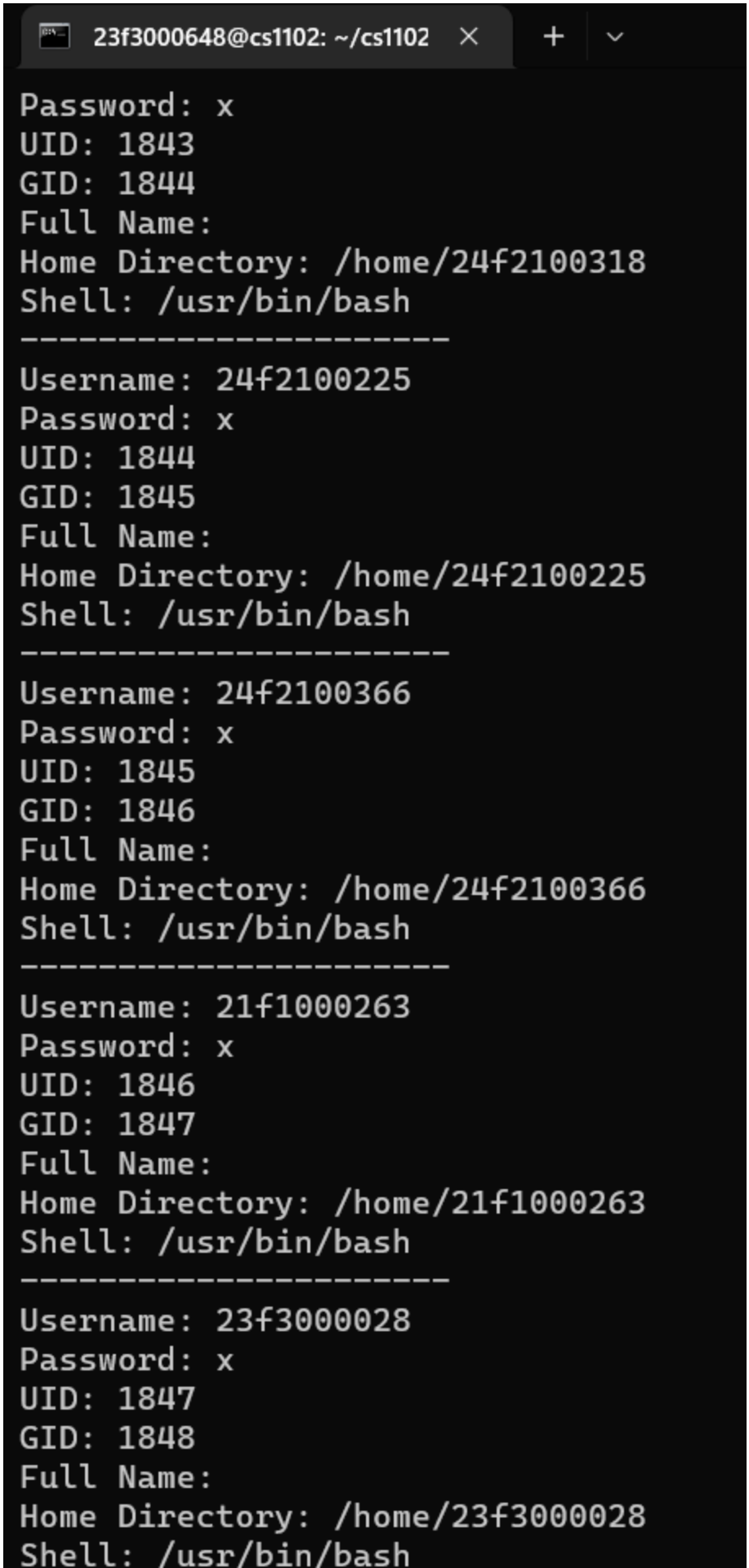
Achieve the same result as above using associative and indexed arrays.
Indexed arrays



```
23f3000648@cs1102: ~/cs1102
23f3000028:x:1847:1848::/home/23f3000028:/usr/bin/bash
23f3000648@cs1102:~/cs1102/lab3$
23f3000648@cs1102:~/cs1102/lab3$ nano bashindexed.sh
23f3000648@cs1102:~/cs1102/lab3$ nano bashindexed.sh
23f3000648@cs1102:~/cs1102/lab3$ bash bashindexed.sh
Username: root
Password: x
UID: 0
GID: 0
Full name: root
Home Directory: /root
Shell: /bin/bash
------------------------
Username: daemon
Password: x
UID: 1
GID: 1
Full name: daemon
Home Directory: /usr/sbin
Shell: /usr/sbin/nologin
------------------------
Username: bin
Password: x
UID: 2
GID: 2
Full name: bin
Home Directory: /bin
Shell: /usr/sbin/nologin
------------------------
Username: sys
Password: x
UID: 3
GID: 3
Full name: sys
Home Directory: /dev
Shell: /usr/sbin/nologin
------------------------
Username: sync
Password: x
UID: 4
```

associative arrays




23f3000648@cs1102: ~/cs1102
Password: x
UID: 1843
GID: 1844
Full Name:
Home Directory: /home/24f2100318
Shell: /usr/bin/bash

Username: 24f2100225
Password: x
UID: 1844
GID: 1845
Full Name:
Home Directory: /home/24f2100225
Shell: /usr/bin/bash

Username: 24f2100366
Password: x
UID: 1845
GID: 1846
Full Name:
Home Directory: /home/24f2100366
Shell: /usr/bin/bash

Username: 21f1000263
Password: x
UID: 1846
GID: 1847
Full Name:
Home Directory: /home/21f1000263
Shell: /usr/bin/bash

Username: 23f3000028
Password: x
UID: 1847
GID: 1848
Full Name:
Home Directory: /home/23f3000028
Shell: /usr/bin/bash

23f3000648@cs1102: ~/cs1102

GNU nano 6.2 bashassociative.sh

```
#!/bin/bash

# Loop through each line of the /etc/passwd file
while read -r line
do
    # Use indexed array to split the line based on the ':' delimiter
    IFS=":" read -r username password uid gid fullname home shell <<< "$line"

    # Create an associative array to store the values with meaningful keys
    declare -A user_info
    user_info["Username"]=$username
    user_info["Password"]=$password
    user_info["UID"]=$uid
    user_info["GID"]=$gid
    user_info["Full Name"]=$fullname
    user_info["Home Directory"]=$home
    user_info["Shell"]=$shell

    # Print the values from the associative array
    echo "Username: ${user_info["Username"]}"
    echo "Password: ${user_info["Password"]}"
    echo "UID: ${user_info["UID"]}"
    echo "GID: ${user_info["GID"]}"
    echo "Full Name: ${user_info["Full Name"]}"
    echo "Home Directory: ${user_info["Home Directory"]}"
    echo "Shell: ${user_info["Shell"]}"
    echo "-----------------------"
done < /etc/passwd
```

3.4.1 Creating Numerous Files with Complex Directory Structure

23f3000648@cs1102: ~/cs1102

```
23f3000648@cs1102:~/cs1102$ for ((i=1; i<=1000; i++)); do
> folder=$(( (i-1 / 25 + 1))
> mkdir -p folder$folder
> echo "$i.txt" > folder$folder/$i.txt
> done
```

```
23f3000648@cs1102:~/cs1102/lab3.4$ ls
folder1   folder12  folder15  folder18  folder20  folder23  folder26  folder29  folder31  folder34  folder37  folder4   folder6  folder9
folder10  folder13  folder16  folder19  folder21  folder24  folder27  folder3   folder32  folder35  folder38  folder40  folder7
folder11  folder14  folder17  folder2   folder22  folder25  folder28  folder30  folder33  folder36  folder39  folder5   folder8
23f3000648@cs1102:~/cs1102/lab3.4$
```

3.4.2 Printing Content of Files

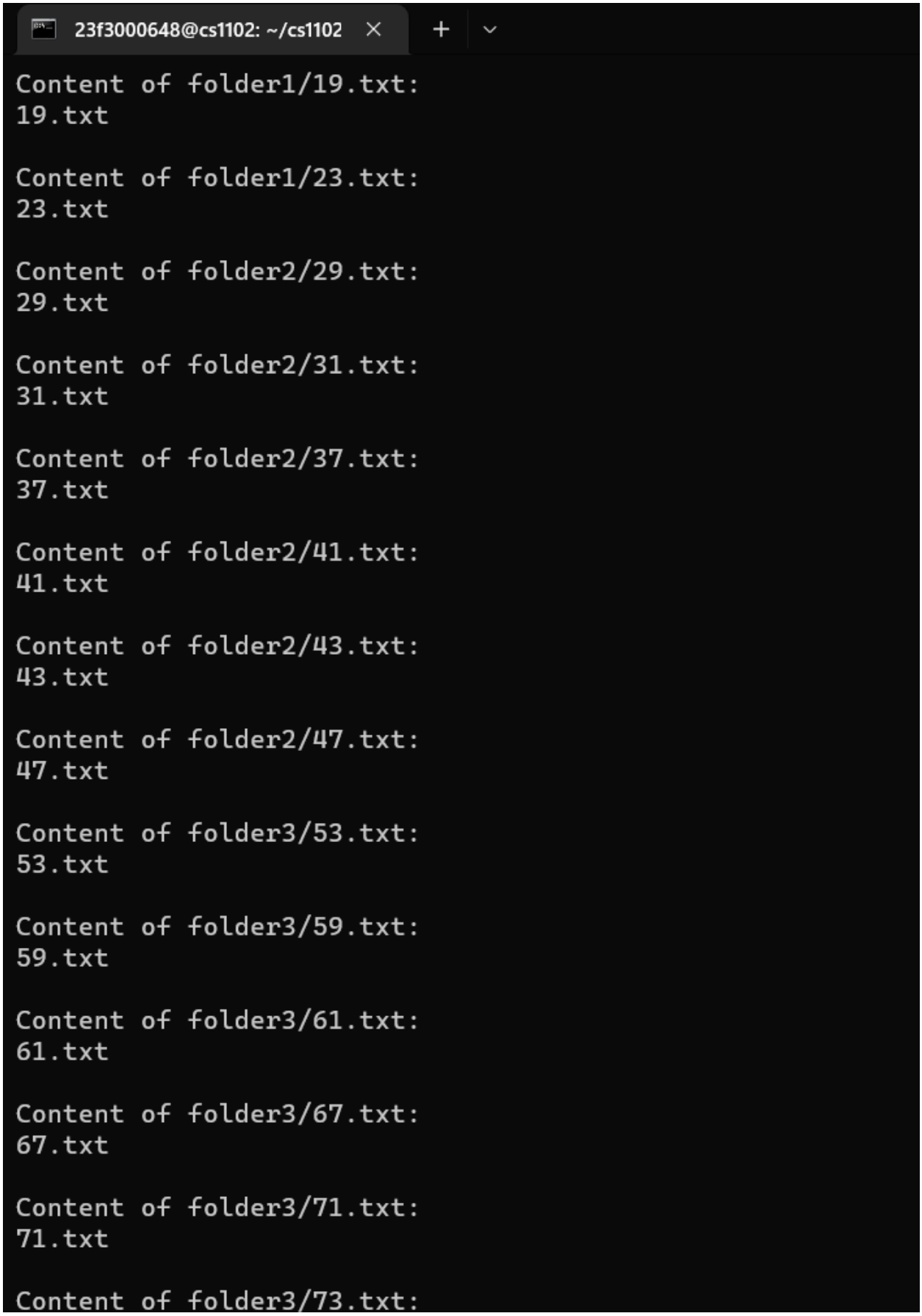
Print the content of files whose names are prime numbers.

```
23f3000648@cs1102: ~/cs1102
23f3000648@cs1102:~/cs1102/lab3.4$
23f3000648@cs1102:~/cs1102/lab3.4$ nano printcont.sh
23f3000648@cs1102:~/cs1102/lab3.4$ bash printcont.sh
```

Bash script for printing

```
23f3000648@cs1102: ~/cs1102
  GNU nano 6.2
for ((i=2; i<=1000; i++)); do
    is_prime=1
    for ((j=2; j*j<=i; j++)); do
        if ((i % j == 0)); then
            is_prime=0
            break
        fi
    done

    if ((is_prime)); then
        file_path="folder$(( (i - 1) / 25 + 1 ))/$i.txt"
        if [[ -f "$file_path" ]]; then
            echo "Content of $file_path:"
            cat "$file_path"
            echo # Print a blank line for readability
        fi
    fi
done
```

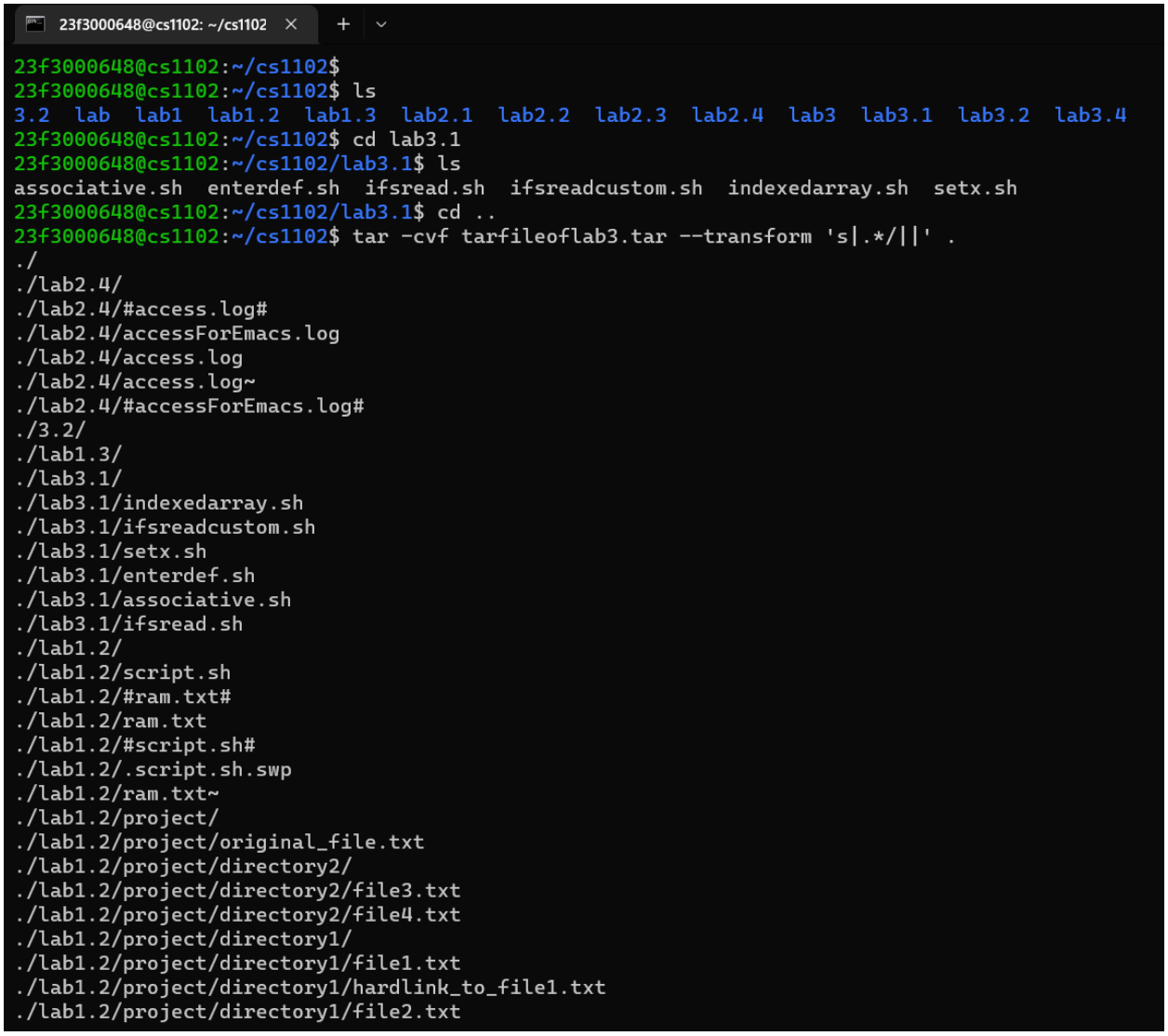
```
23f3000648@cs1102:~/cs1102/lab5.4$ bash printC
Content of folder1/2.txt:
2.txt

Content of folder1/3.txt:
3.txt

Content of folder1/5.txt:
5.txt

Content of folder1/7.txt:
7.txt

Content of folder1/11.txt:
11.txt

Content of folder1/13.txt:
13.txt

Content of folder1/17.txt:
17.txt

Content of folder1/19.txt:
19.txt

Content of folder1/23.txt:
23.txt

Content of folder2/29.txt:
29.txt

Content of folder2/31.txt:
31.txt

Content of folder2/37.txt:
37.txt

Content of folder2/41.txt:
```


23f3000648@cs1102: ~/cs1102
Content of folder1/19.txt:
19.txt
Content of folder1/23.txt:
23.txt
Content of folder2/29.txt:
29.txt
Content of folder2/31.txt:
31.txt
Content of folder2/37.txt:
37.txt
Content of folder2/41.txt:
41.txt
Content of folder2/43.txt:
43.txt
Content of folder2/47.txt:
47.txt
Content of folder3/53.txt:
53.txt
Content of folder3/59.txt:
59.txt
Content of folder3/61.txt:
61.txt
Content of folder3/67.txt:
67.txt
Content of folder3/71.txt:
71.txt
Content of folder3/73.txt:

3.5 Using Zip and Tar Functionality

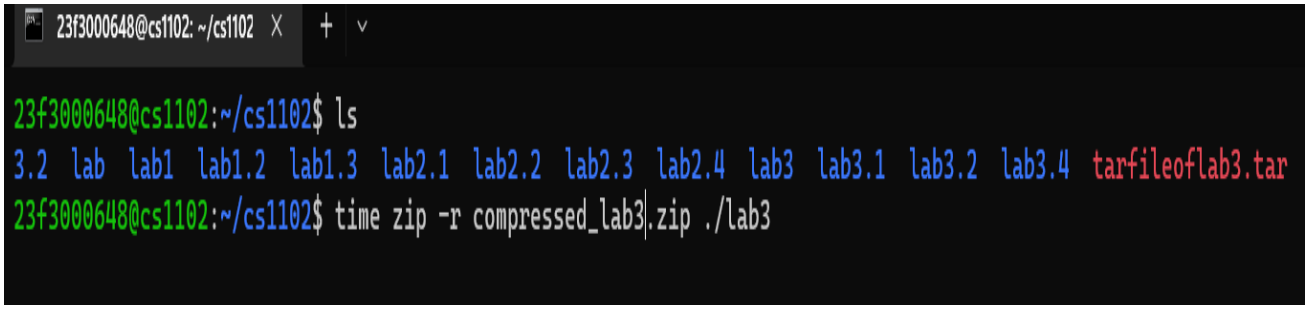
1. Flattening Directory with Tar

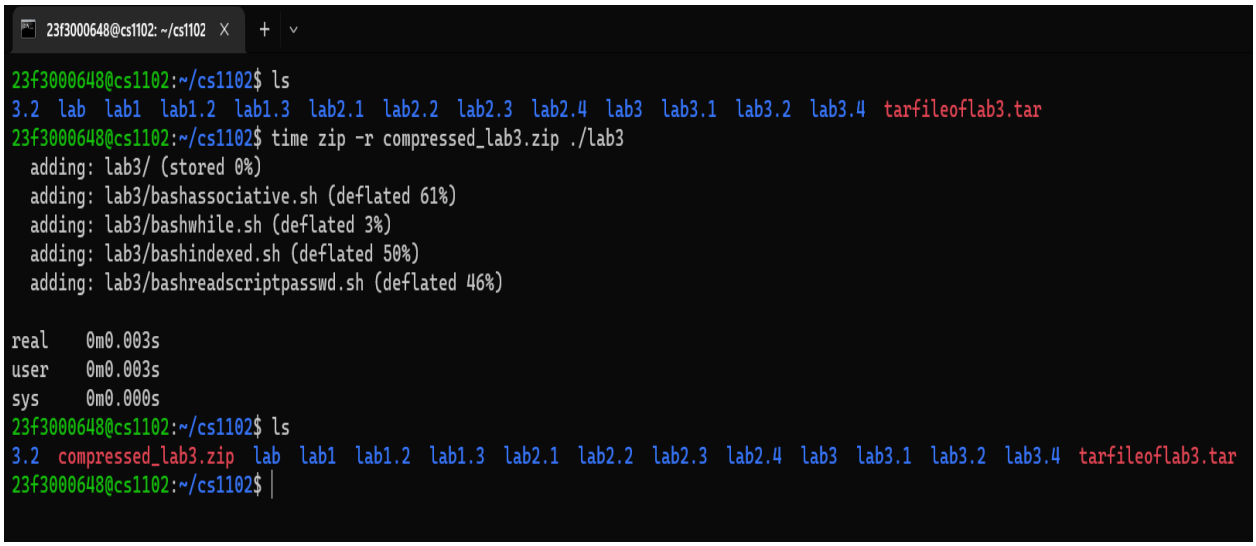
Use the tar command to flatten the directory structure.



Timing Zip Operations

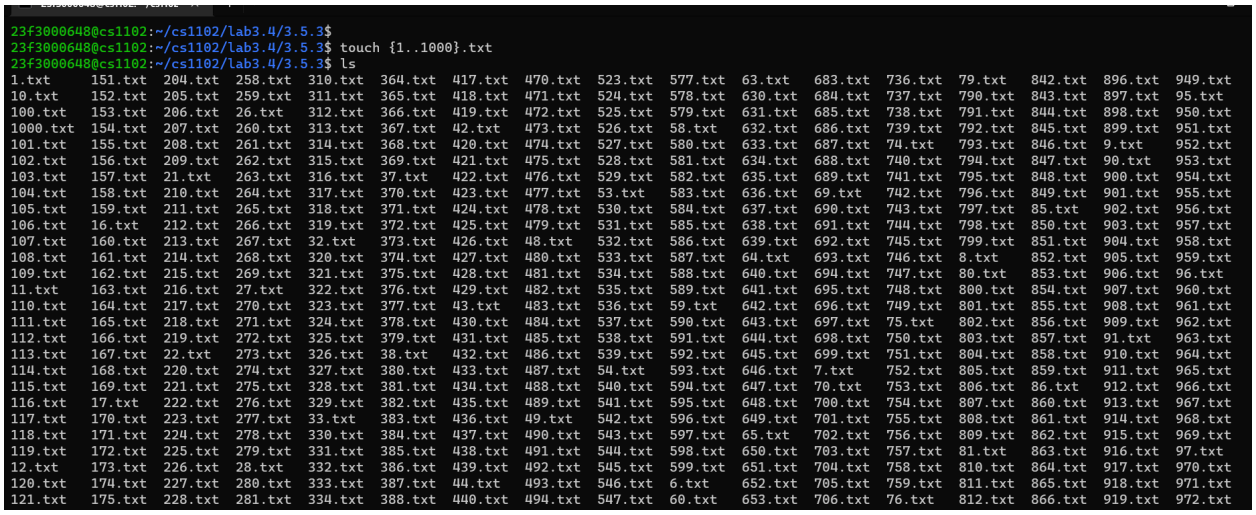
Use the zip command to compress files and folders, noting the time taken for the operation.



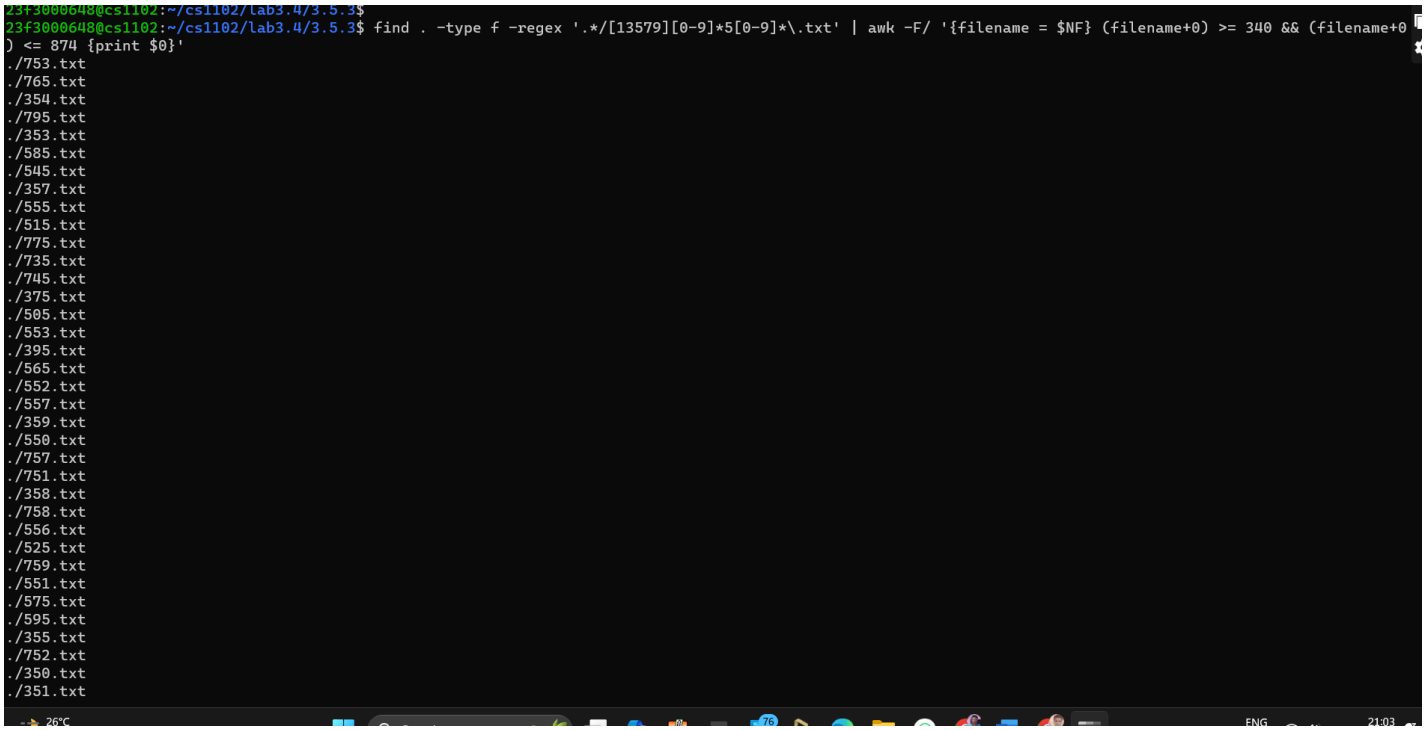


Finding Files with Find Command

Use the find command to print the content of files above 340 and below 874 if the filename is odd and has the digit 5 in it.



```
23f3000648@cs1102:~/cs1102/lab3.4/3.5.3$
23f3000648@cs1102:~/cs1102/lab3.4/3.5.3$ touch {1..1000}.txt
23f3000648@cs1102:~/cs1102/lab3.4/3.5.3$ ls
1.txt     151.txt  204.txt  258.txt  310.txt  364.txt  417.txt  470.txt  523.txt  577.txt  63.txt   683.txt  736.txt  79.txt   842.txt  896.txt  949.txt
10.txt    152.txt  205.txt  259.txt  311.txt  365.txt  418.txt  471.txt  524.txt  578.txt  630.txt  684.txt  737.txt  790.txt  843.txt  897.txt  95.txt
100.txt   153.txt  206.txt  26.txt   312.txt  366.txt  419.txt  472.txt  525.txt  579.txt  631.txt  685.txt  738.txt  791.txt  844.txt  898.txt  950.txt
1000.txt  154.txt  207.txt  260.txt  313.txt  367.txt  42.txt   473.txt  526.txt  58.txt   632.txt  686.txt  739.txt  792.txt  845.txt  899.txt  951.txt
101.txt   155.txt  208.txt  261.txt  314.txt  368.txt  420.txt  474.txt  527.txt  580.txt  633.txt  687.txt  74.txt   793.txt  846.txt  9.txt    952.txt
102.txt   156.txt  209.txt  262.txt  315.txt  369.txt  421.txt  475.txt  528.txt  581.txt  634.txt  688.txt  740.txt  794.txt  847.txt  90.txt   953.txt
103.txt   157.txt  21.txt   263.txt  316.txt  37.txt   422.txt  476.txt  529.txt  582.txt  635.txt  689.txt  741.txt  795.txt  848.txt  900.txt  954.txt
104.txt   158.txt  210.txt  264.txt  317.txt  370.txt  423.txt  477.txt  53.txt   583.txt  636.txt  69.txt   742.txt  796.txt  849.txt  901.txt  955.txt
105.txt   159.txt  211.txt  265.txt  318.txt  371.txt  424.txt  478.txt  530.txt  584.txt  637.txt  690.txt  743.txt  797.txt  85.txt   902.txt  956.txt
106.txt   16.txt   212.txt  266.txt  319.txt  372.txt  425.txt  479.txt  531.txt  585.txt  638.txt  691.txt  744.txt  798.txt  850.txt  903.txt  957.txt
107.txt   160.txt  213.txt  267.txt  32.txt   373.txt  426.txt  48.txt   532.txt  586.txt  639.txt  692.txt  745.txt  799.txt  851.txt  904.txt  958.txt
108.txt   161.txt  214.txt  268.txt  320.txt  374.txt  427.txt  480.txt  533.txt  587.txt  64.txt   693.txt  746.txt  8.txt    852.txt  905.txt  959.txt
109.txt   162.txt  215.txt  269.txt  321.txt  375.txt  428.txt  481.txt  534.txt  588.txt  640.txt  694.txt  747.txt  80.txt   853.txt  906.txt  96.txt
11.txt    163.txt  216.txt  27.txt   322.txt  376.txt  429.txt  482.txt  535.txt  589.txt  641.txt  695.txt  748.txt  800.txt  854.txt  907.txt  960.txt
110.txt   164.txt  217.txt  270.txt  323.txt  377.txt  43.txt   483.txt  536.txt  59.txt   642.txt  696.txt  749.txt  801.txt  855.txt  908.txt  961.txt
111.txt   165.txt  218.txt  271.txt  324.txt  378.txt  430.txt  484.txt  537.txt  590.txt  643.txt  697.txt  75.txt   802.txt  856.txt  909.txt  962.txt
112.txt   166.txt  219.txt  272.txt  325.txt  379.txt  431.txt  485.txt  538.txt  591.txt  644.txt  698.txt  750.txt  803.txt  857.txt  91.txt   963.txt
113.txt   167.txt  22.txt   273.txt  326.txt  38.txt   432.txt  486.txt  539.txt  592.txt  645.txt  699.txt  751.txt  804.txt  858.txt  910.txt  964.txt
114.txt   168.txt  220.txt  274.txt  327.txt  380.txt  433.txt  487.txt  54.txt   593.txt  646.txt  7.txt    752.txt  805.txt  859.txt  911.txt  965.txt
115.txt   169.txt  221.txt  275.txt  328.txt  381.txt  434.txt  488.txt  540.txt  594.txt  647.txt  70.txt   753.txt  806.txt  86.txt   912.txt  966.txt
116.txt   17.txt   222.txt  276.txt  329.txt  382.txt  435.txt  489.txt  541.txt  595.txt  648.txt  700.txt  754.txt  807.txt  860.txt  913.txt  967.txt
117.txt   170.txt  223.txt  277.txt  33.txt   383.txt  436.txt  49.txt   542.txt  596.txt  649.txt  701.txt  755.txt  808.txt  861.txt  914.txt  968.txt
118.txt   171.txt  224.txt  278.txt  330.txt  384.txt  437.txt  490.txt  543.txt  597.txt  65.txt   702.txt  756.txt  809.txt  862.txt  915.txt  969.txt
119.txt   172.txt  225.txt  279.txt  331.txt  385.txt  438.txt  491.txt  544.txt  598.txt  650.txt  703.txt  757.txt  81.txt   863.txt  916.txt  97.txt
12.txt    173.txt  226.txt  28.txt   332.txt  386.txt  439.txt  492.txt  545.txt  599.txt  651.txt  704.txt  758.txt  810.txt  864.txt  917.txt  970.txt
120.txt   174.txt  227.txt  280.txt  333.txt  387.txt  44.txt   493.txt  546.txt  6.txt    652.txt  705.txt  759.txt  811.txt  865.txt  918.txt  971.txt
121.txt   175.txt  228.txt  281.txt  334.txt  388.txt  440.txt  494.txt  547.txt  60.txt   653.txt  706.txt  76.txt   812.txt  866.txt  919.txt  972.txt
```



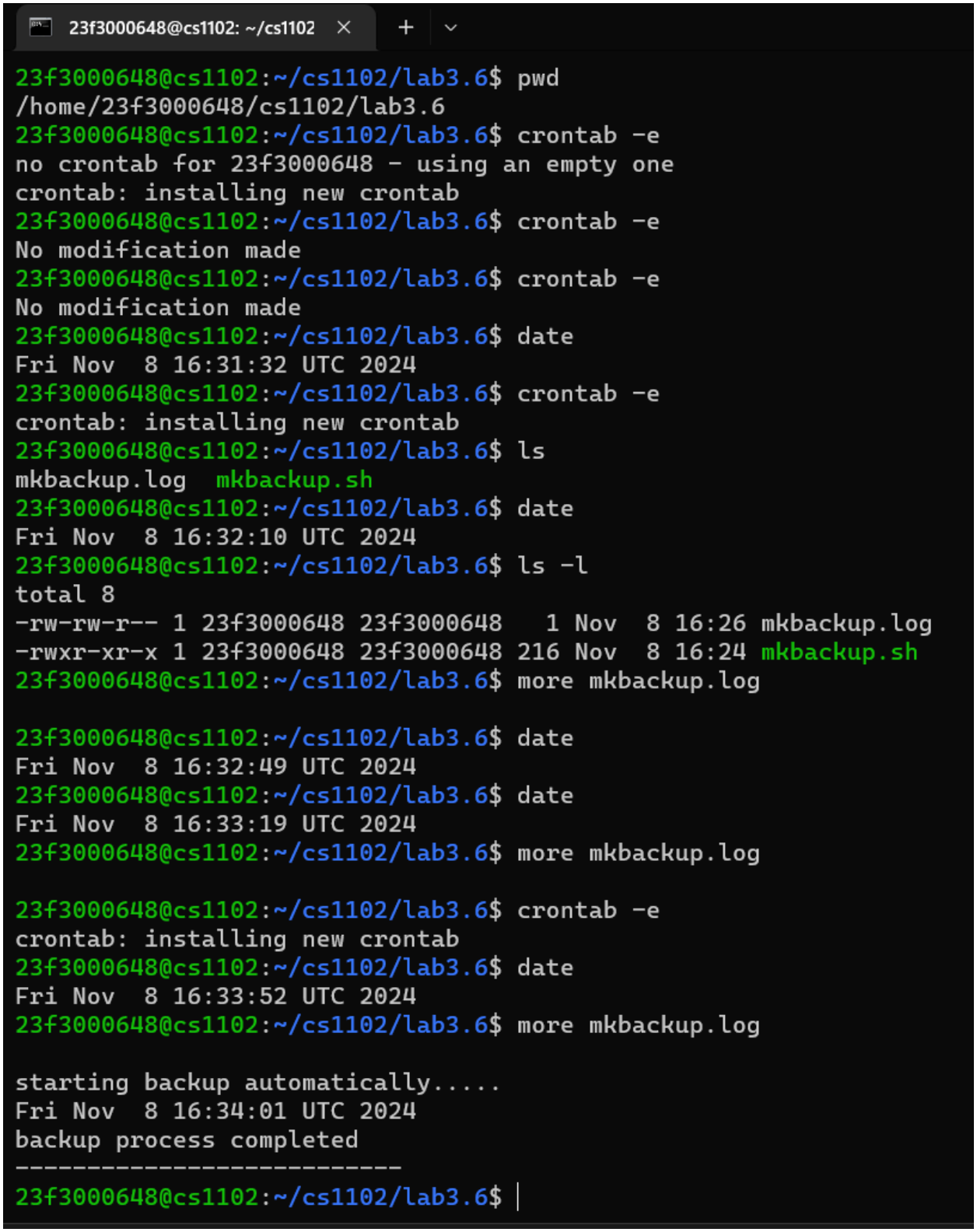
```
23f3000648@cs1102:~/cs1102/lab3.4/3.5.3$
23f3000648@cs1102:~/cs1102/lab3.4/3.5.3$ find . -type f -regex '.*/[13579][0-9]*5[0-9]*\.txt' | awk -F/ '{filename = $NF} (filename+0) >= 340 && (filename+0
) <= 874 {print $0}'
./753.txt
./765.txt
./354.txt
./795.txt
./353.txt
./585.txt
./545.txt
./357.txt
./555.txt
./515.txt
./775.txt
./735.txt
./745.txt
./375.txt
./505.txt
./553.txt
./395.txt
./565.txt
./552.txt
./557.txt
./359.txt
./550.txt
./757.txt
./751.txt
./358.txt
./758.txt
./556.txt
./525.txt
./759.txt
./551.txt
./575.txt
./595.txt
./355.txt
./752.txt
./350.txt
./351.txt
```

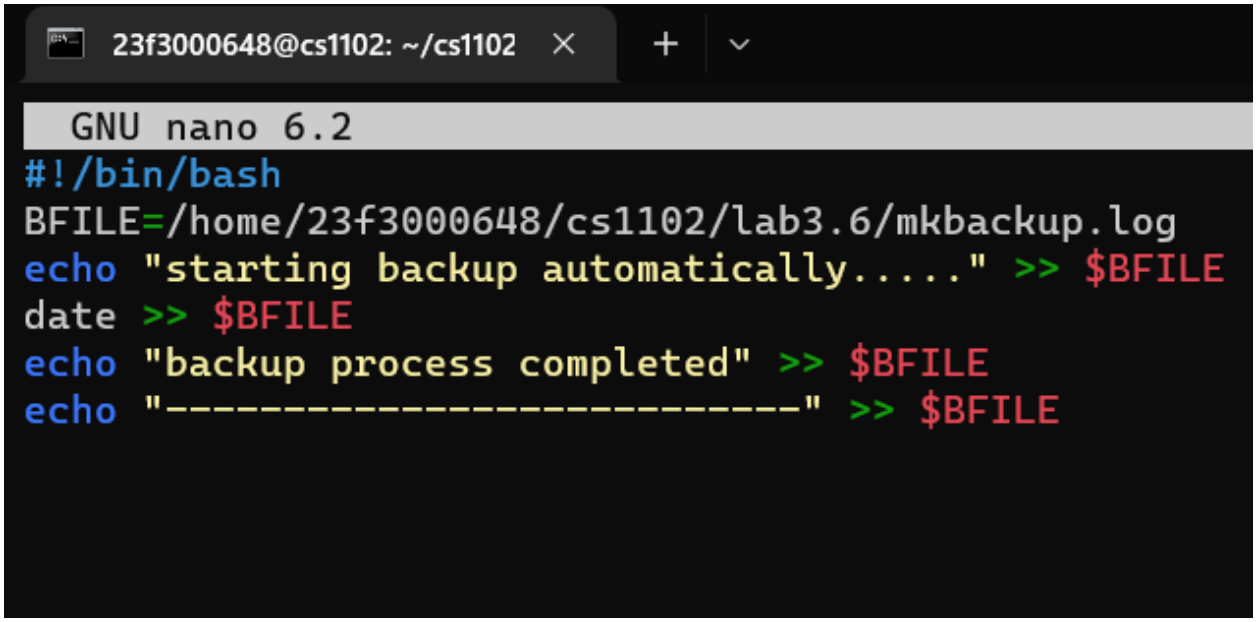
3.6 Setting up Backup Script Using Make Utility and Crontab

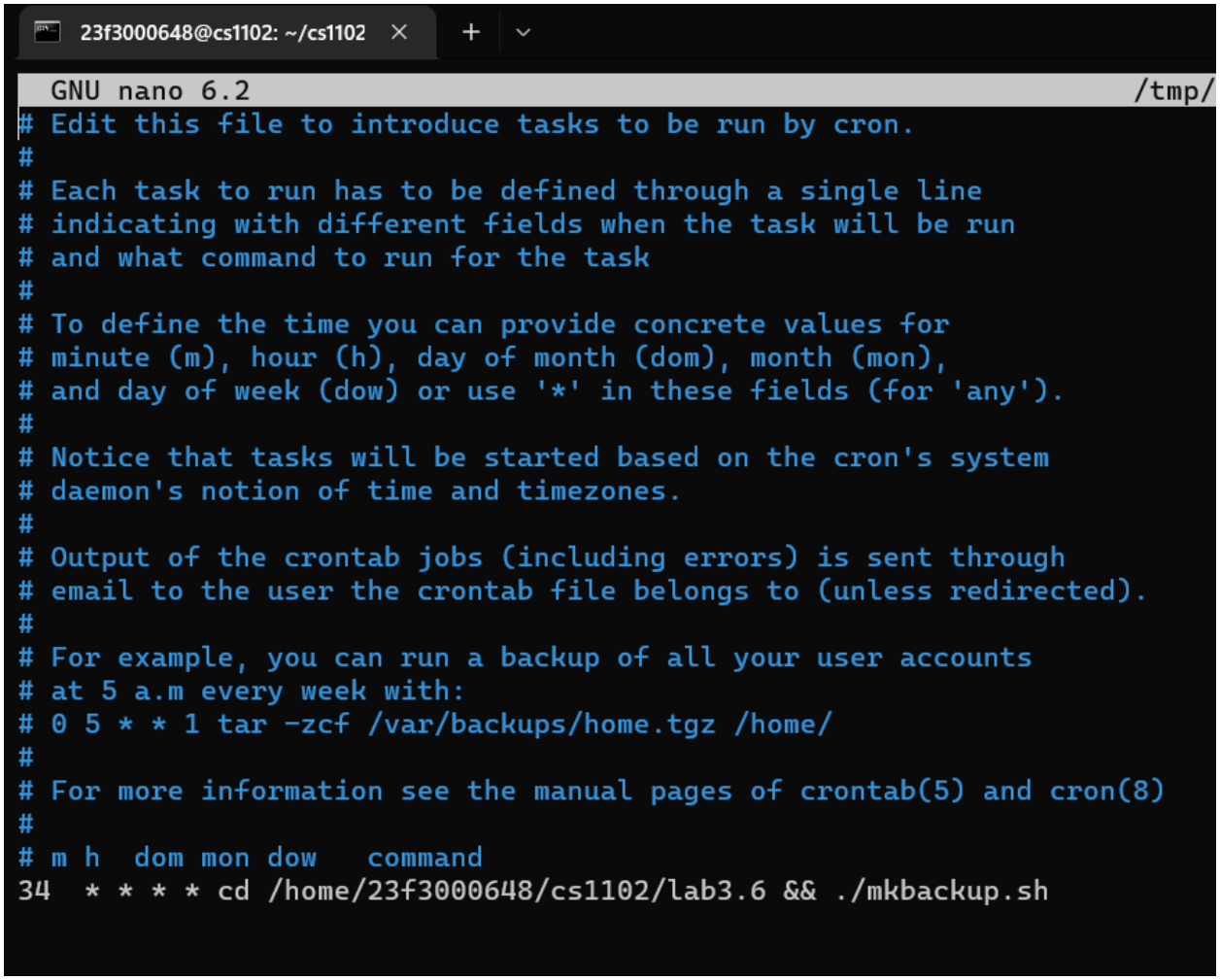
Backup Script

Create and run a backup script using the make utility and execute it using the crontab command. Mimic the backup script created in the lecture.

```
23f3000648@cs1102: ~/cs1102
23f3000648@cs1102:~/cs1102/lab3.6$ ls
mkbackup.log  mkbackup.sh
23f3000648@cs1102:~/cs1102/lab3.6$ nano mkbackup.sh
23f3000648@cs1102:~/cs1102/lab3.6$ more mkbackup.sh
#!/bin/bash
BFILE=/home/23f3000648/cs1102/lab3.6/mkbackup.log
echo "starting backup automatically....." >> $BFILE
date >> $BFILE
echo "backup process completed" >> $BFILE
echo "--------------------------" >> $BFILE
23f3000648@cs1102:~/cs1102/lab3.6$ crontab -e
no crontab for 23f3000648 - using an empty one

Select an editor.  To change later, run 'select-editor'.
  1. /bin/nano        <---- easiest
  2. /usr/bin/vim.basic
  3. /usr/bin/mcedit
  4. /usr/bin/vim.tiny
  5. /usr/bin/emacs
  6. /bin/ed

Choose 1-6 [1]: 1
No modification made
23f3000648@cs1102:~/cs1102/lab3.6$ chmod 755 mkbackup.sh
23f3000648@cs1102:~/cs1102/lab3.6$ ./mkbackup.sh
23f3000648@cs1102:~/cs1102/lab3.6$ cat mkbackup.
cat: mkbackup.: No such file or directory
23f3000648@cs1102:~/cs1102/lab3.6$ cat mkbackup.log
starting backup automatically.....
Fri Nov  8 16:25:54 UTC 2024
backup process completed
--------------------------
23f3000648@cs1102:~/cs1102/lab3.6$ nano mkbackup.log
23f3000648@cs1102:~/cs1102/lab3.6$ crontab -e
no crontab for 23f3000648 - using an empty one
No modification made
23f3000648@cs1102:~/cs1102/lab3.6$ pwd
/home/23f3000648/cs1102/lab3.6
23f3000648@cs1102:~/cs1102/lab3.6$
```


23f3000648@cs1102: ~/cs1102
23f3000648@cs1102:~/cs1102/lab3.6$ pwd
/home/23f3000648/cs1102/lab3.6
23f3000648@cs1102:~/cs1102/lab3.6$ crontab -e
no crontab for 23f3000648 - using an empty one
crontab: installing new crontab
23f3000648@cs1102:~/cs1102/lab3.6$ crontab -e
No modification made
23f3000648@cs1102:~/cs1102/lab3.6$ crontab -e
No modification made
23f3000648@cs1102:~/cs1102/lab3.6$ date
Fri Nov 8 16:31:32 UTC 2024
23f3000648@cs1102:~/cs1102/lab3.6$ crontab -e
crontab: installing new crontab
23f3000648@cs1102:~/cs1102/lab3.6$ ls
mkbackup.log mkbackup.sh
23f3000648@cs1102:~/cs1102/lab3.6$ date
Fri Nov 8 16:32:10 UTC 2024
23f3000648@cs1102:~/cs1102/lab3.6$ ls -l
total 8
-rw-rw-r-- 1 23f3000648 23f3000648 1 Nov 8 16:26 mkbackup.log
-rwxr-xr-x 1 23f3000648 23f3000648 216 Nov 8 16:24 mkbackup.sh
23f3000648@cs1102:~/cs1102/lab3.6$ more mkbackup.log
23f3000648@cs1102:~/cs1102/lab3.6$ date
Fri Nov 8 16:32:49 UTC 2024
23f3000648@cs1102:~/cs1102/lab3.6$ date
Fri Nov 8 16:33:19 UTC 2024
23f3000648@cs1102:~/cs1102/lab3.6$ more mkbackup.log
23f3000648@cs1102:~/cs1102/lab3.6$ crontab -e
crontab: installing new crontab
23f3000648@cs1102:~/cs1102/lab3.6$ date
Fri Nov 8 16:33:52 UTC 2024
23f3000648@cs1102:~/cs1102/lab3.6$ more mkbackup.log
starting backup automatically.....
Fri Nov 8 16:34:01 UTC 2024
backup process completed
23f3000648@cs1102:~/cs1102/lab3.6$

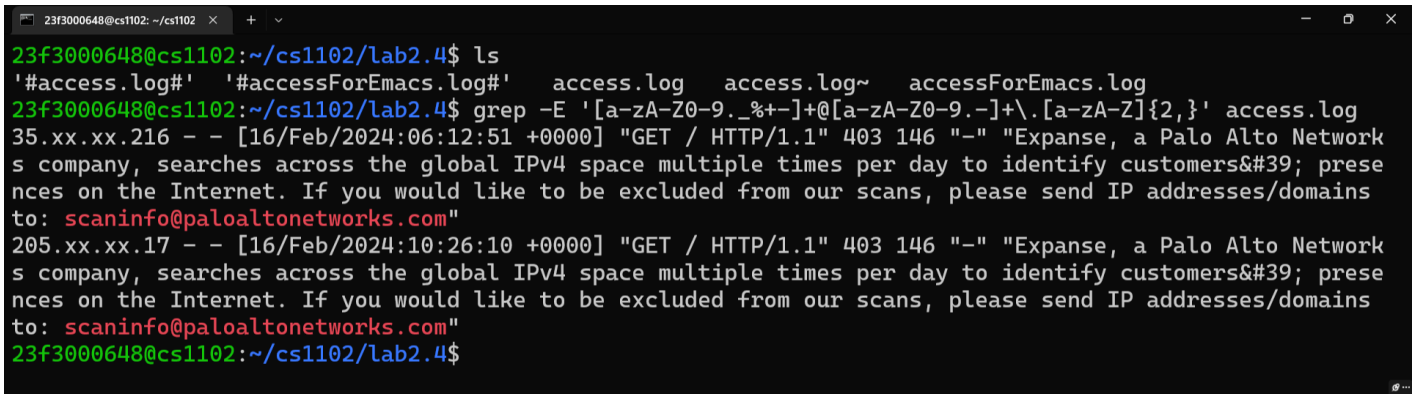

23f3000648@cs1102: ~/cs1102
GNU nano 6.2
#!/bin/bash
BFILE=/home/23f3000648/cs1102/lab3.6/mkbackup.log
echo "starting backup automatically....." >> $BFILE
date >> $BFILE
echo "backup process completed" >> $BFILE
echo "--------------------------" >> $BFILE


23f3000648@cs1102: ~/cs1102
GNU nano 6.2 /tmp/
Edit this file to introduce tasks to be run by cron.
#
Each task to run has to be defined through a single line
indicating with different fields when the task will be run
and what command to run for the task
#
To define the time you can provide concrete values for
minute (m), hour (h), day of month (dom), month (mon),
and day of week (dow) or use '*' in these fields (for 'any').
#
Notice that tasks will be started based on the cron's system
daemon's notion of time and timezones.
#
Output of the crontab jobs (including errors) is sent through
email to the user the crontab file belongs to (unless redirected).
#
For example, you can run a backup of all your user accounts
at 5 a.m every week with:
0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
For more information see the manual pages of crontab(5) and cron(8)
#
m h dom mon dow command
34 * * * * cd /home/23f3000648/cs1102/lab3.6 && ./mkbackup.sh

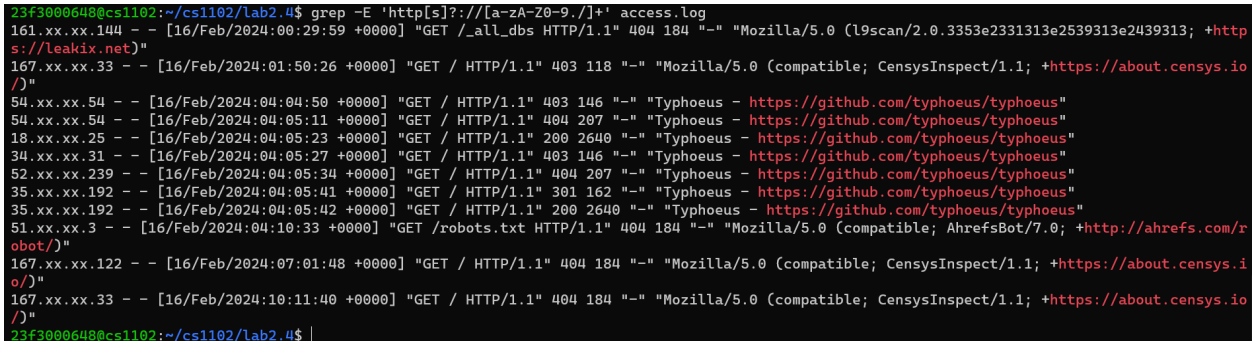
4.1 Regular Expressions

1.Matching a Specific Pattern:

Task: Match all email addresses in a text file.



Match all URLs in a text file.

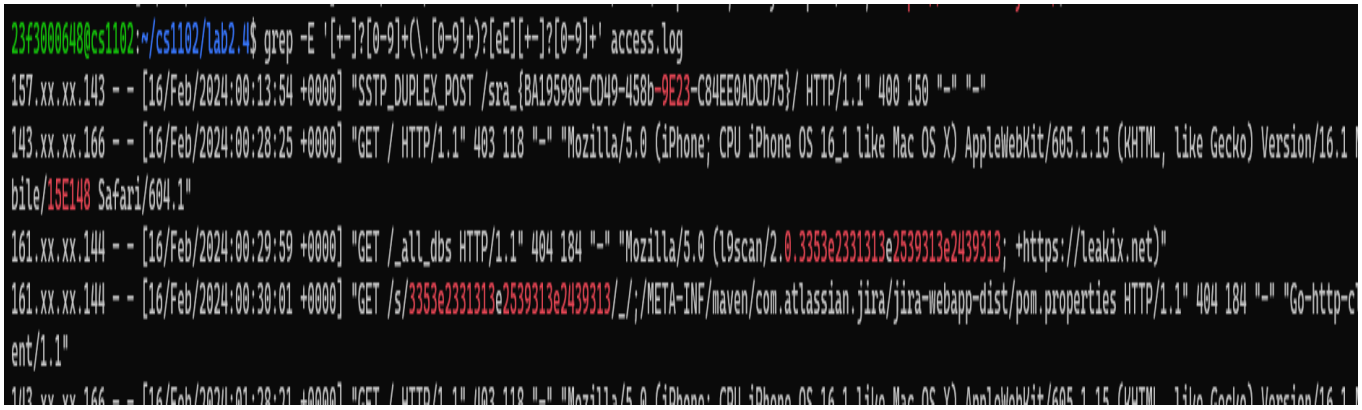


2. Matching Digits:

Task: Match all phone numbers in a text file.

```
23f3000648@cs1102: ~/cs1102
23f3000648@cs1102:~/cs1102/lab2.4$ touch phonenumbers.txt
23f3000648@cs1102:~/cs1102/lab2.4$ nano phonenumbers.txt
23f3000648@cs1102:~/cs1102/lab2.4$ grep -E '\b[0-9]{10}\b' access.log
23f3000648@cs1102:~/cs1102/lab2.4$ grep -E '\b[0-9]{10}\b' phonenumbers.txt
Ram Tripathi: 1234567890
Shyam Tripathi: 9919910168
Rajesh Tripathi: 9919912168
23f3000648@cs1102:~/cs1102/lab2.4$
```

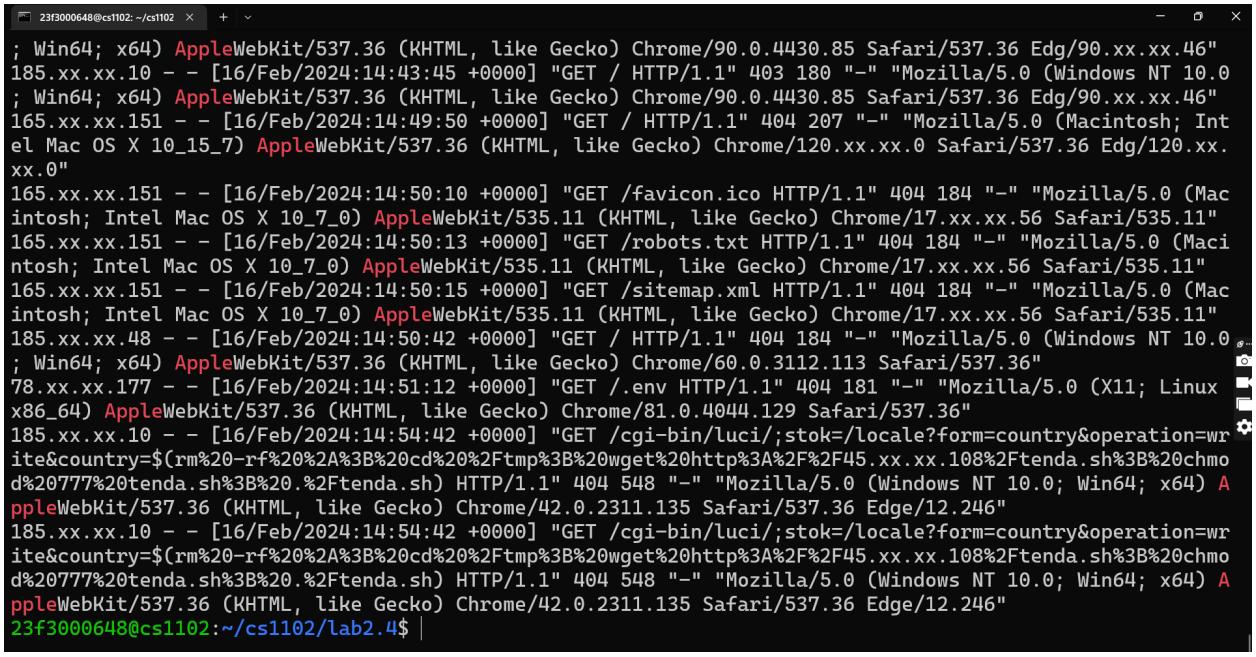
Match all numbers in scientific notation



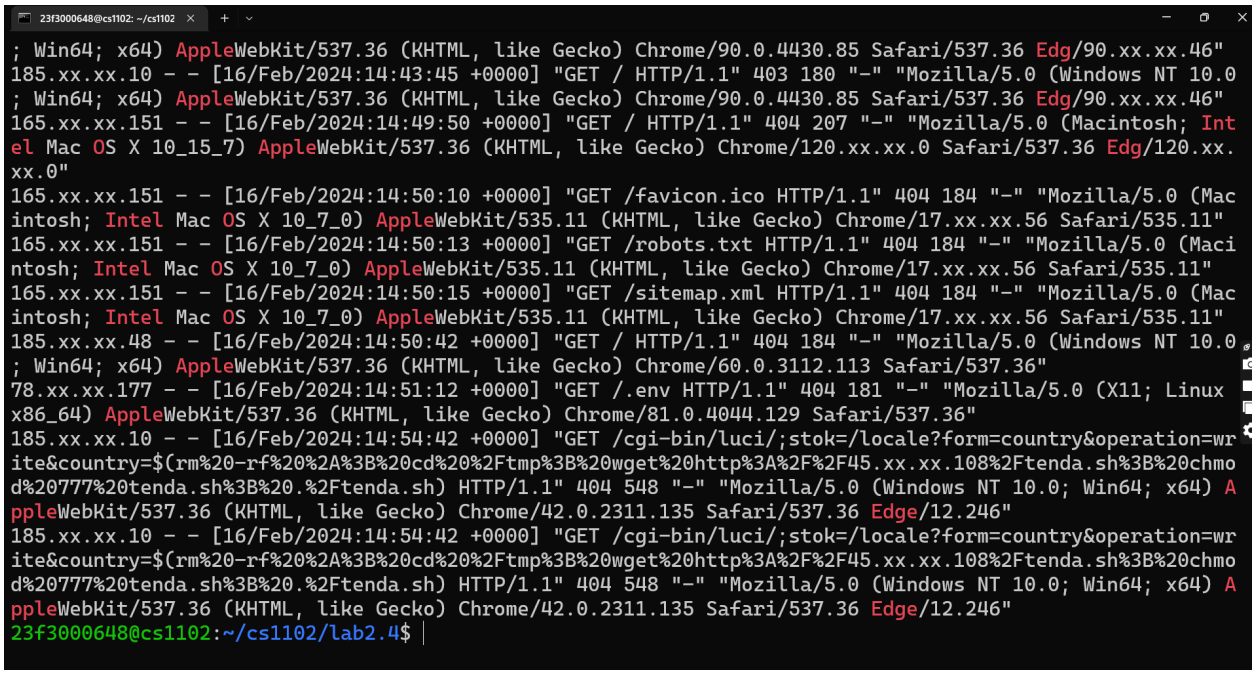
Matching Words Starting with a Specific Letter:

Task: Match all words starting with a letter e.g. 'apple' in a text file.

```
23f3000648@cs1102: ~/cs1102
23f3000648@cs1102:~/cs1102/lab2.4$ grep -E '\b[aA][a-z]*' access.log
```

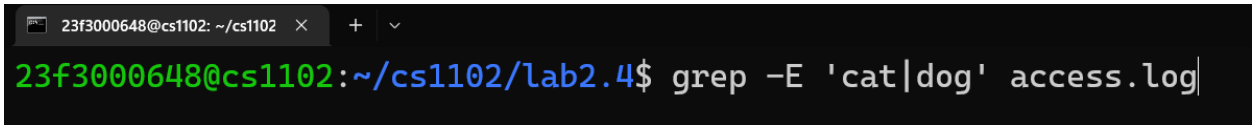


Match all words starting with a vowel

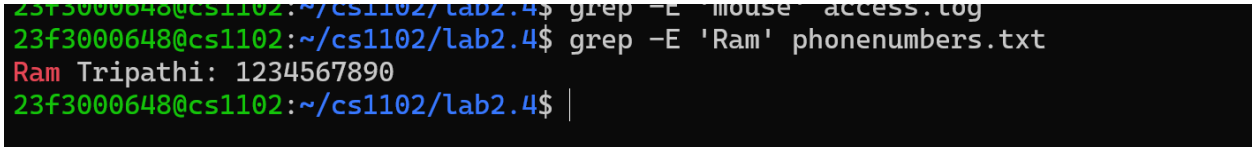


Matching Specific Characters:

Task: Match all lines containing either of a word e.g. 'cat' or 'dog'.

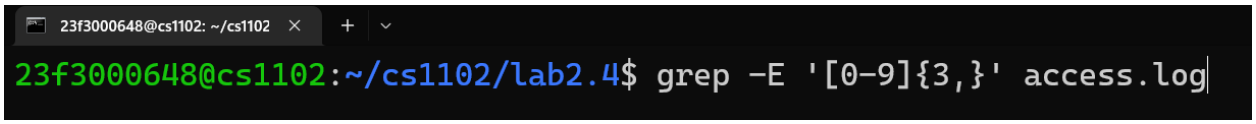


Task: Match all lines containing word ‘Ram’

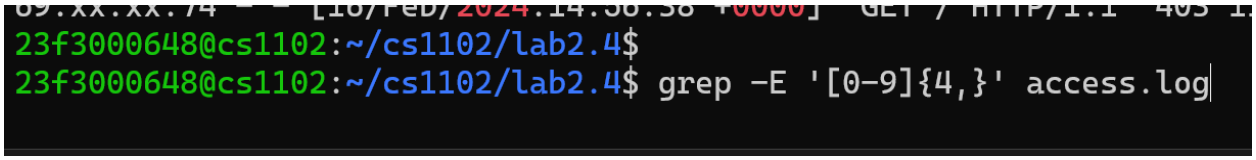


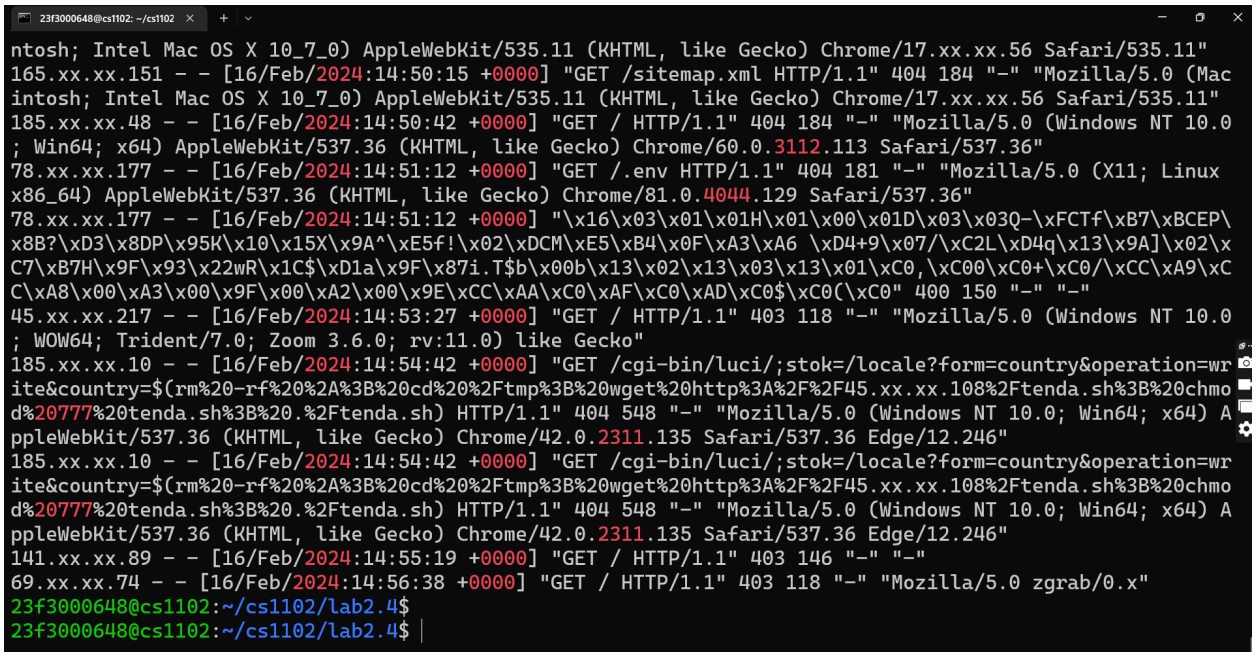
Quantifiers:
Syntax: *, +, ?, {n}, {n,}, {n,m}
Purpose: Specifies the number of repetitions for the preceding element.

Task: Match all lines containing at least n digits e.g. 3 consecutive digits.



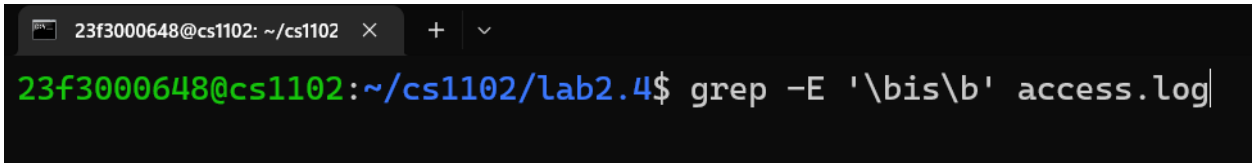
Match all lines containing at least 4 digits

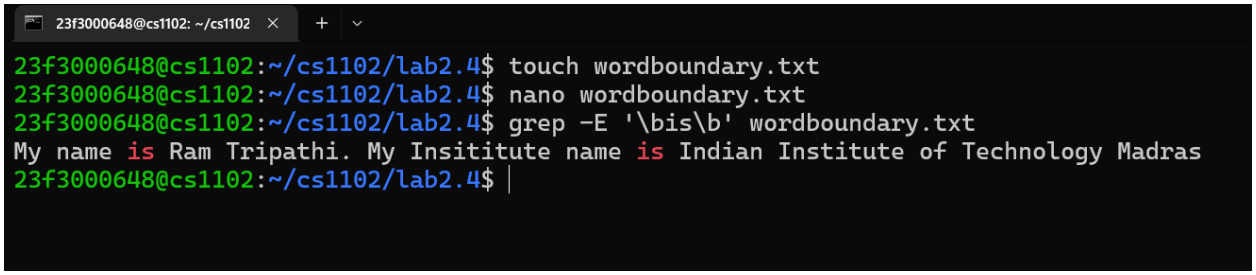




Matching Word Boundaries:

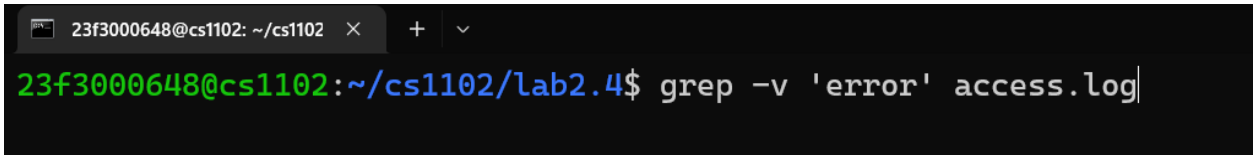
Task: Match all occurrences of a whole word e.g. 'is' using word boundary

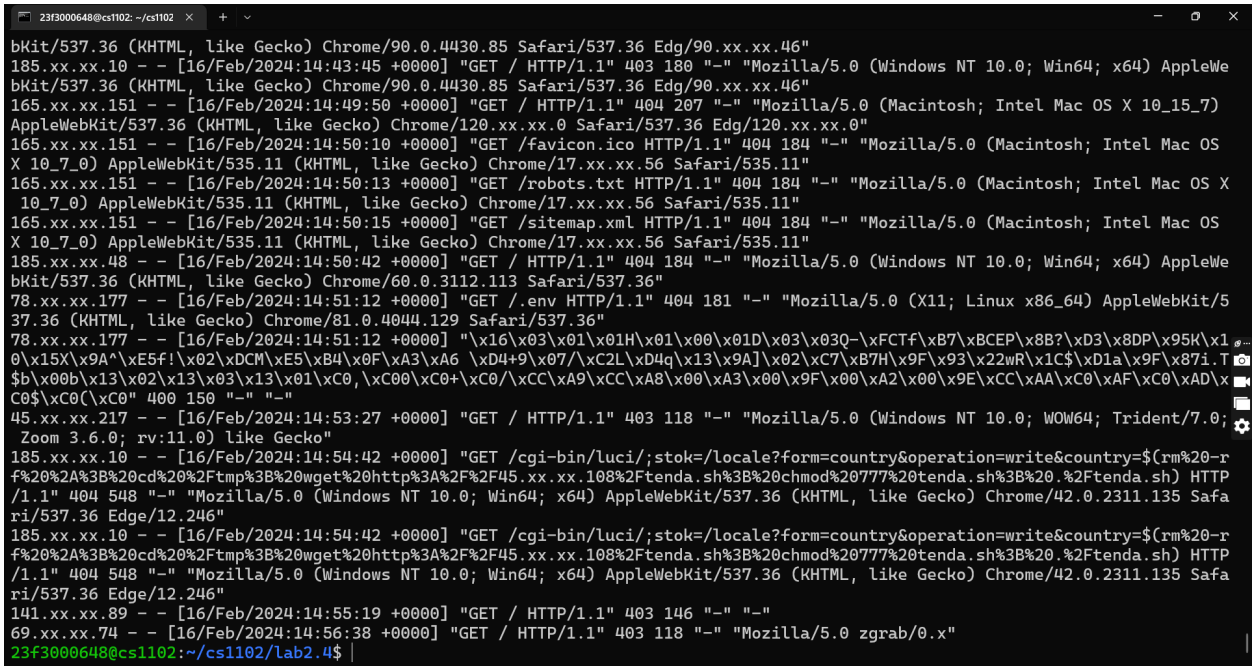




Negation: (bonus)

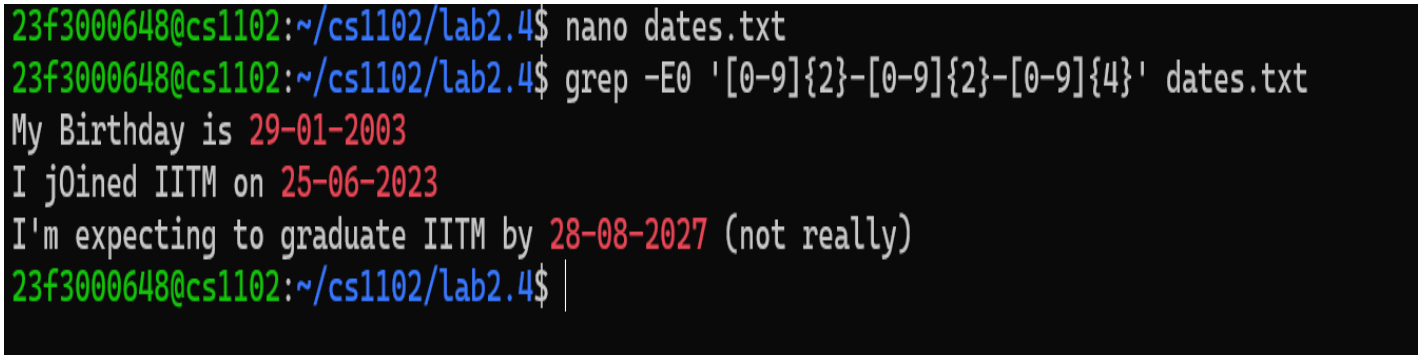
Task: Match all lines not containing a word e.g. the word 'error' in an authlog file.



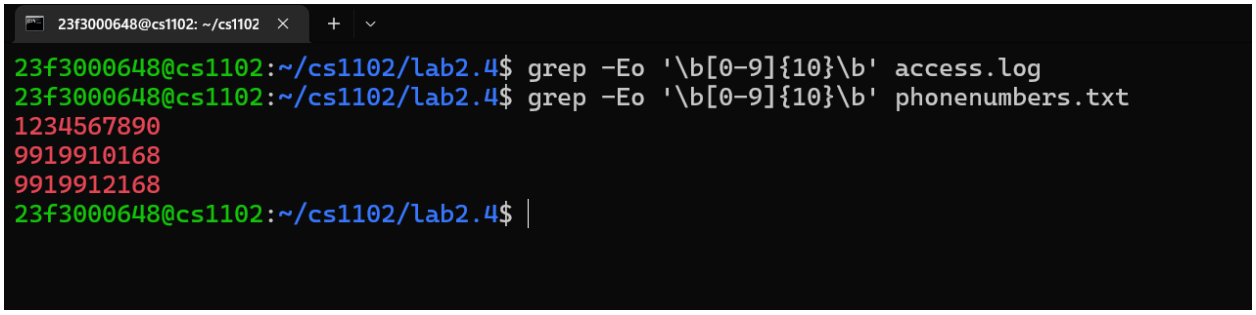


Capturing Groups:

Task: Match and capture the date components (year, month, day) from a date string. You can do the same for ip address or mac address or phone number using groups concept.



Phone numbers matched



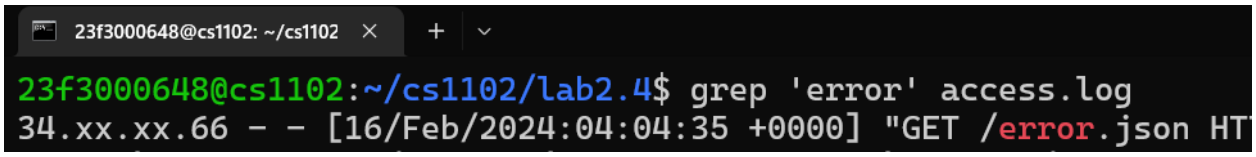
4.2 Grep

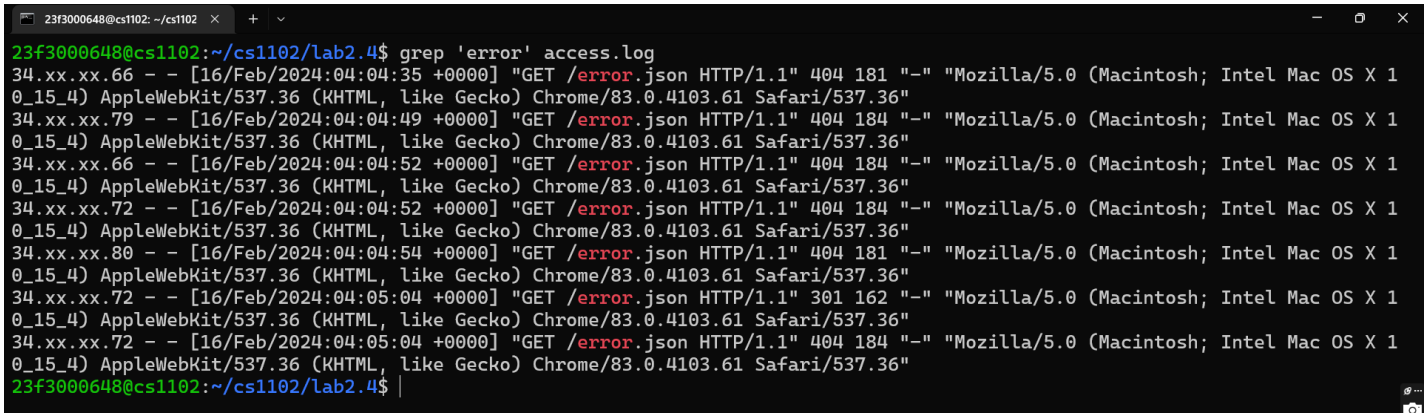
grep Workbook

For the following do the task and create one more similar task there by compiling at least 2 examples/use-cases for each of the sub-titles.

Basic Pattern Search:

Task: Search for the word 'error' in a log file.

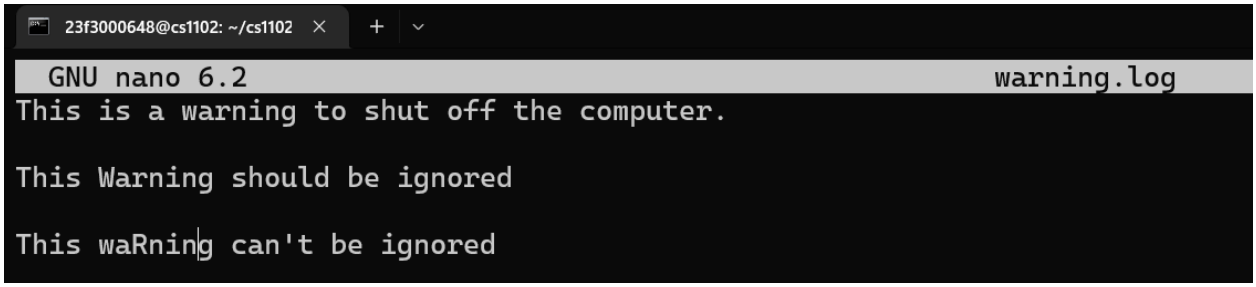




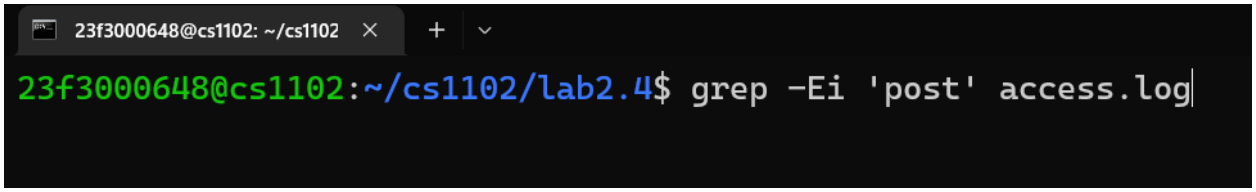
Search for the word 'ram' in a log file.

Case Insensitive Search:

Task: Search for the word 'warning' case insensitively in a log file

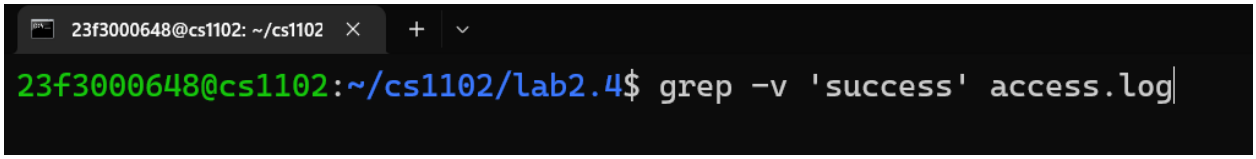


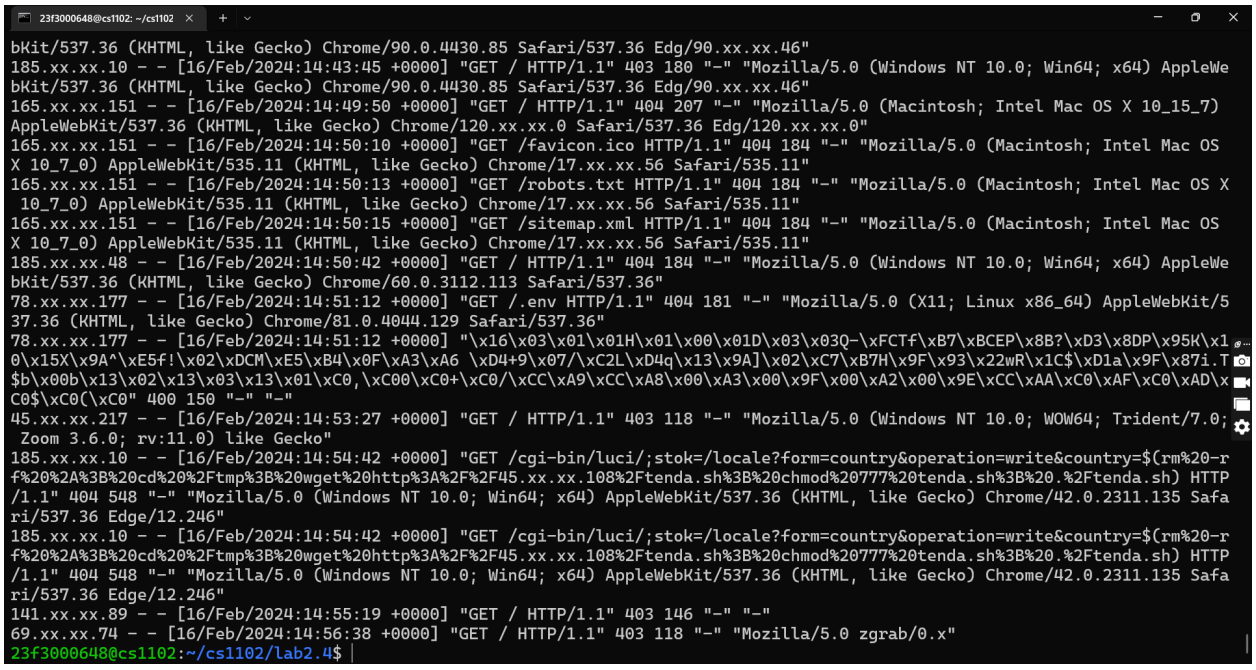
Search for the word 'post' case insensitively in a log file



Inverted Match:

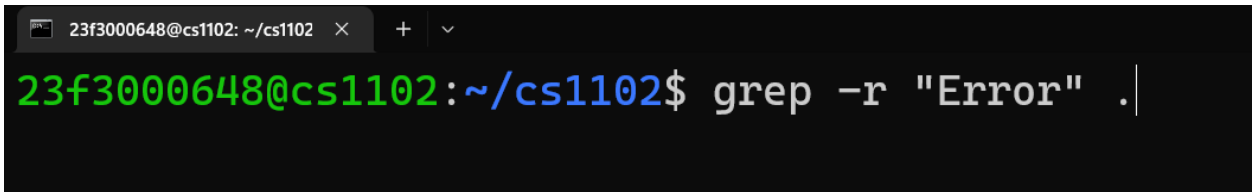
Task: List all lines not containing the word 'success'





Recursive Search:

Task: Search for a pattern recursively in all files within a directory. (you will need to create a directory structure with some content in files in the directory structure. You can use the Lab 3 for loop activity to do the task.)

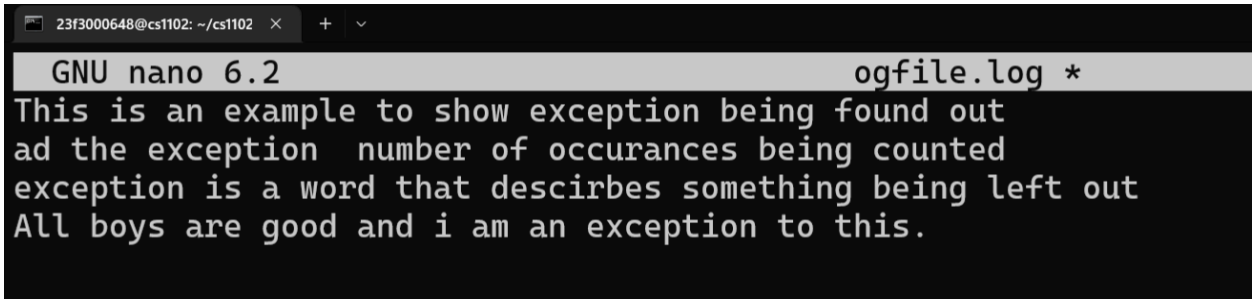


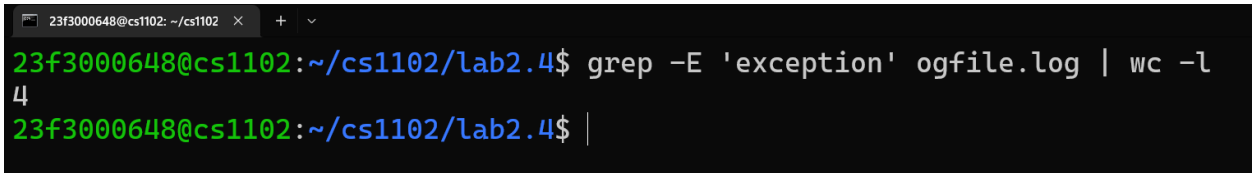

```
23f3000648@cs1102:~/cs1102$ grep -r "Error" .
./lab2.4/#access.log#:178.xx.xx.171 - - [16/Feb/2024:04:04:39 +0000] "GET /docs/cplugError.html/ HTTP/1.1" 404 207 "-" "curl/
7.54.0"
./lab2.4/#access.log#:178.xx.xx.171 - - [16/Feb/2024:04:04:39 +0000] "GET /docs/cplugError.html/ HTTP/1.1" 404 146 "-" "curl/
7.54.0"
./lab2.4/accessForEmacs.log:178.79.139.171 - - [16/Feb/2024:04:04:39 +0000] "GET /docs/cplugError.html/ HTTP/1.1" 404 207 "-"
 "curl/7.54.0"
./lab2.4/accessForEmacs.log:178.79.139.171 - - [16/Feb/2024:04:04:39 +0000] "GET /docs/cplugError.html/ HTTP/1.1" 404 146 "-"
 "curl/7.54.0"
./lab2.4/access.log:178.xx.xx.171 - - [16/Feb/2024:04:04:39 +0000] "GET /docs/cplugError.html/ HTTP/1.1" 404 207 "-" "curl/7.
54.0"
./lab2.4/access.log:178.xx.xx.171 - - [16/Feb/2024:04:04:39 +0000] "GET /docs/cplugError.html/ HTTP/1.1" 404 146 "-" "curl/7.
54.0"
./lab2.4/access.log~:178.xx.xx.171 - - [16/Feb/2024:04:04:39 +0000] "GET /docs/cplugErro
.html/ HTTP/1.1" 404 207 "-" "curl/7.54.0"
./lab2.4/access.log~:178.xx.xx.171 - - [16/Feb/2024:04:04:39 +0000] "GET /docs/cplugErro
.html/ HTTP/1.1" 404 146 "-" "curl/7.54.0"
./lab2.4/#accessForEmacs.log#:178.79.139.171 - - [16/Feb/2024:04:04:39 +0000] "GET /docs/cplugError.html/ HTTP/1.1" 404 207 "
-" "curl/7.54.0"
./lab2.4/#accessForEmacs.log#:178.79.139.171 - - [16/Feb/2024:04:04:39 +0000] "GET /docs/cplugError.html/ HTTP/1.1" 404 146 "
-" "curl/7.54.0"
grep: ./tarfileoflab3.tar: binary file matches
./lab2.3/access.log:178.79.139.171 - - [16/Feb/2024:04:04:39 +0000] "GET /docs/cplugErro
.html/ HTTP/1.1" 404 207 "-" "curl/7.54.0"
./lab2.3/access.log:178.79.139.171 - - [16/Feb/2024:04:04:39 +0000] "GET /docs/cplugErro
.html/ HTTP/1.1" 404 146 "-" "curl/7.54.0"
./lab4/task1/file1.txt:Error: File not found
```

Search Recursively for the word Ram

Counting Matches:

Task: Count the number of occurrences of 'exception' in a log file.





```
23f3000648@cs1102:~/cs1102/lab2.4$ grep -E 'exception' accessForEmacs.log | wc -l
0
23f3000648@cs1102:~/cs1102/lab2.4$
```

Displaying Line Numbers:

Task: Show line numbers for all occurrences of 'timeout' in a file.

```
23f3000648@cs1102:~/cs1102/lab2.4$ echo -e "The operation completed successfully.\nA timeout o
ccurred during the process.\nTimeout is set to 30 seconds by default.\nNo issues were detected
." > sample.txt
23f3000648@cs1102:~/cs1102/lab2.4$ grep -n "timeout" sample.txt
2:A timeout occurred during the process.
23f3000648@cs1102:~/cs1102/lab2.4$ cat sample.txt
The operation completed successfully.
A timeout occurred during the process.
Timeout is set to 30 seconds by default.
No issues were detected.
23f3000648@cs1102:~/cs1102/lab2.4$ grep -n "timeout" sample.txt
2:A timeout occurred during the process.
23f3000648@cs1102:~/cs1102/lab2.4$
```

Finding number of occurrences of word Delhi

```
23f3000648@cs1102:~/cs1102/lab2.4$ grep -E 'Delhi' delhi | wc -l
```

```
  GNU nano 6.2                                          delhi *
Delhi is National Captial of India
Delhi is an ancient city
Delhi is the most pollued city
Delhi has 10 districts and is also part of NCR region
```

```
23f3000648@cs1102:~/cs1102/lab2.4$ grep -E 'Delhi' delhi | wc -l
4
23f3000648@cs1102:~/cs1102/lab2.4$
```

Search for Whole Words:

Task: Search for the word 'bug' as a whole word in a text file.

```
23f3000648@cs1102:~/cs1102/lab2.4$ grep -w 'bug' bug.txt
bug refersto presence of code in a software that prevents proper functioning. The ENIAC, one o
f the first computers, famously encountered a "bug" that changed history. This "bug" wasn't me
taphorical-it was a literal moth stuck in the machine, disrupting its operation. ENIAC's relia
nce on vacuum tubes made it susceptible to malfunctions, and the discovery of this bug marked
the first recorded instance of debugging. Despite the "bug," ENIAC revolutionized computing, s
howcasing immense speed and accuracy compared to human calculations. Its success paved the way
 for modern machines, but the bug story serves as a reminder of the challenges pioneers faced.
 Today, "debugging" honors this historic ENIAC mishap in computational folklore.
23f3000648@cs1102:~/cs1102/lab2.4$
```

Search for the word 'Electronics' as a whole word in a text file.

```
23f3000648@cs1102:~/cs1102/lab2.4$ grep -w "Electronics" electronics.txt
Electronics have revolutionized the modern world, beginning with foundational innovations like
 the ENIAC. This early computer, driven by vacuum tube electronics, showcased the power of ele
ctronic circuits to process data rapidly. Electronics enabled ENIAC to solve complex calculati
ons previously unmanageable by humans, marking a milestone in computational history. Over time
, advancements in electronics transitioned from vacuum tubes to transistors, drastically impro
ving efficiency. Today, electronics power nearly every aspect of life, from smartphones to spa
cecraft. The ENIAC symbolizes how the field of electronics began shaping technology, highlight
ing a journey of innovation that continues to define the modern digital era.
23f3000648@cs1102:~/cs1102/lab2.4$
```

Regular Expression Search:

Task: Search for all lines starting with 'Error' (case insensitive).

```
23f3000648@cs1102: ~/cs1102
23f3000648@cs1102:~/cs1102/lab2.4$ nano log.txt
23f3000648@cs1102:~/cs1102/lab2.4$ grep -i "^Error" log.txt
Error: File not found
error: Disk full
Error: Timeout occurred
23f3000648@cs1102:~/cs1102/lab2.4$
```

Search for all lines starting with 'Enull' (case insensitive)

```
23f3000648@cs1102: ~/cs1102
23f3000648@cs1102:~/cs1102/lab2.4$ nano ciphers.txt
23f3000648@cs1102:~/cs1102/lab2.4$ grep -i "^Enull" ciphers.txt
eNULL: Cipher not supported
ENull: Incorrect configuration
eNull: Connection failed
23f3000648@cs1102:~/cs1102/lab2.4$
```

4.3 SED

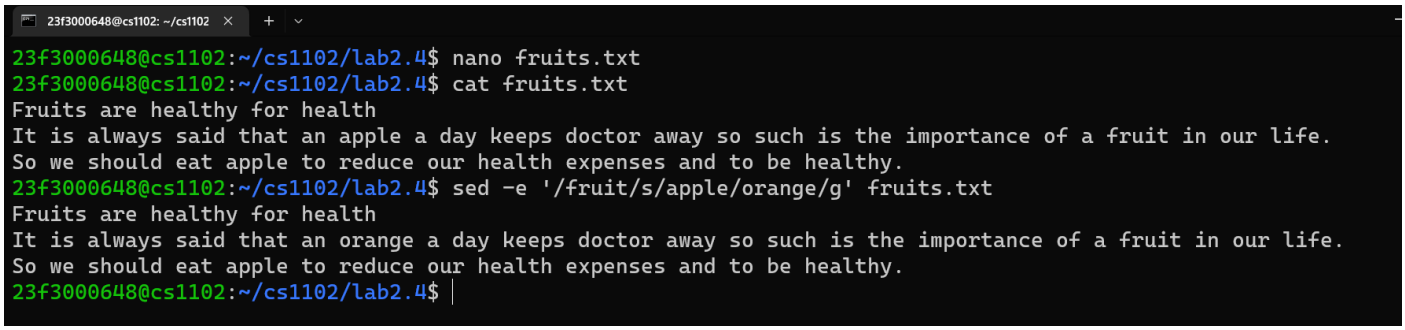
Substitution:

Task: Replace 'old' with 'new' in a file.

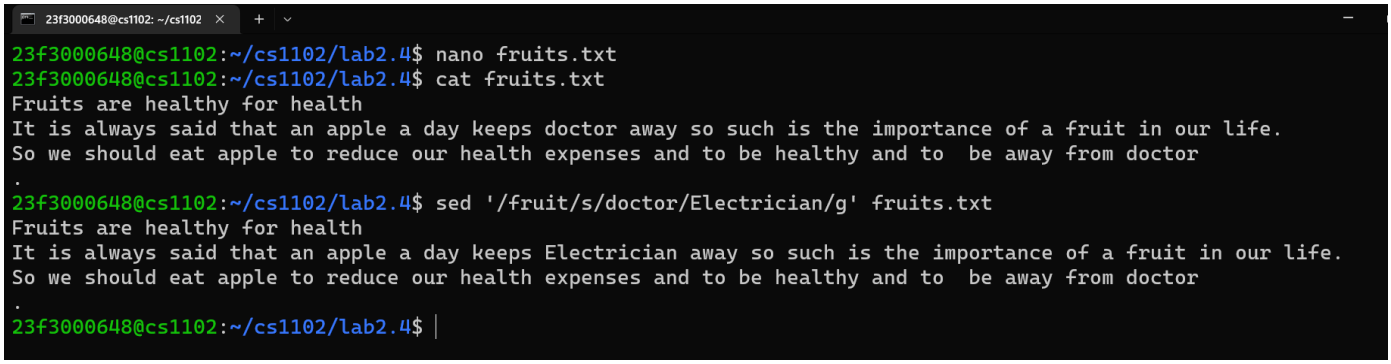
```
23f3000648@cs1102: ~/cs1102
23f3000648@cs1102:~/cs1102/lab2.4$ nano sample.txt
23f3000648@cs1102:~/cs1102/lab2.4$ cat sample.txt
The operation completed successfully.
A timeout occurred during the process.
Timeout is set to 30 seconds by default.
No issues were detected.


The old fashion is making is making its way into new fashion.
23f3000648@cs1102:~/cs1102/lab2.4$ sed -e 's/old/new/g' sample.txt
The operation completed successfully.
A timeout occurred during the process.
Timeout is set to 30 seconds by default.
No issues were detected.


The new fashion is making is making its way into new fashion.
23f3000648@cs1102:~/cs1102/lab2.4$
```

Task: Replace 'seconds' with 'minutes' in a file

Selective Substitution:
Task: Replace 'apple' with 'orange' only in lines containing 'fruit'.

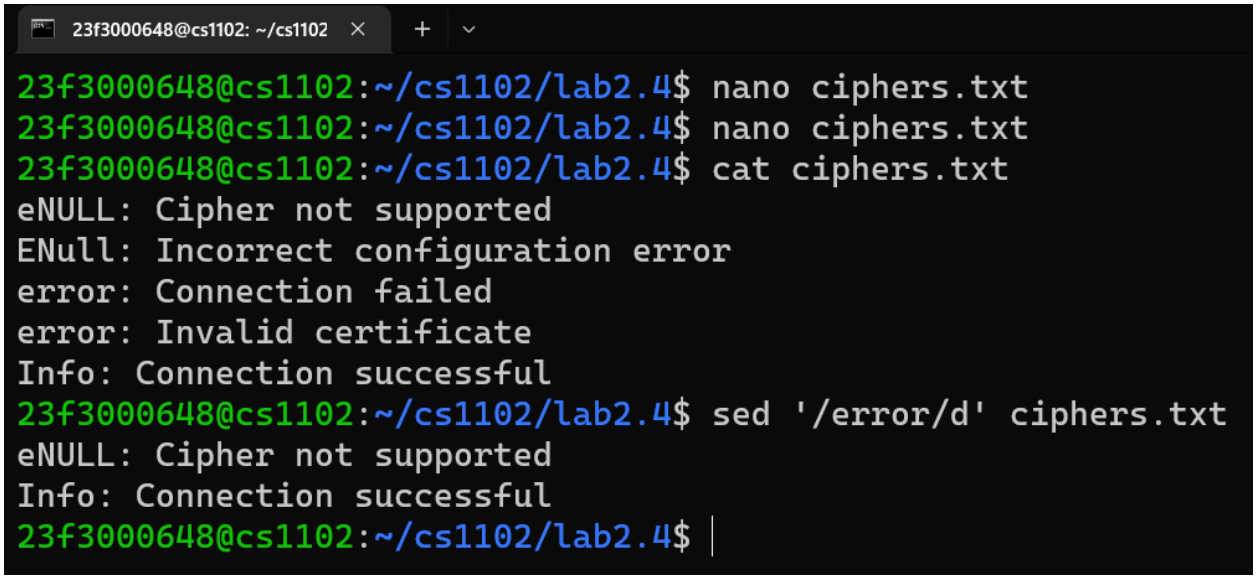


Replace 'doctor' with 'Electrician' only in lines containing 'fruit'.

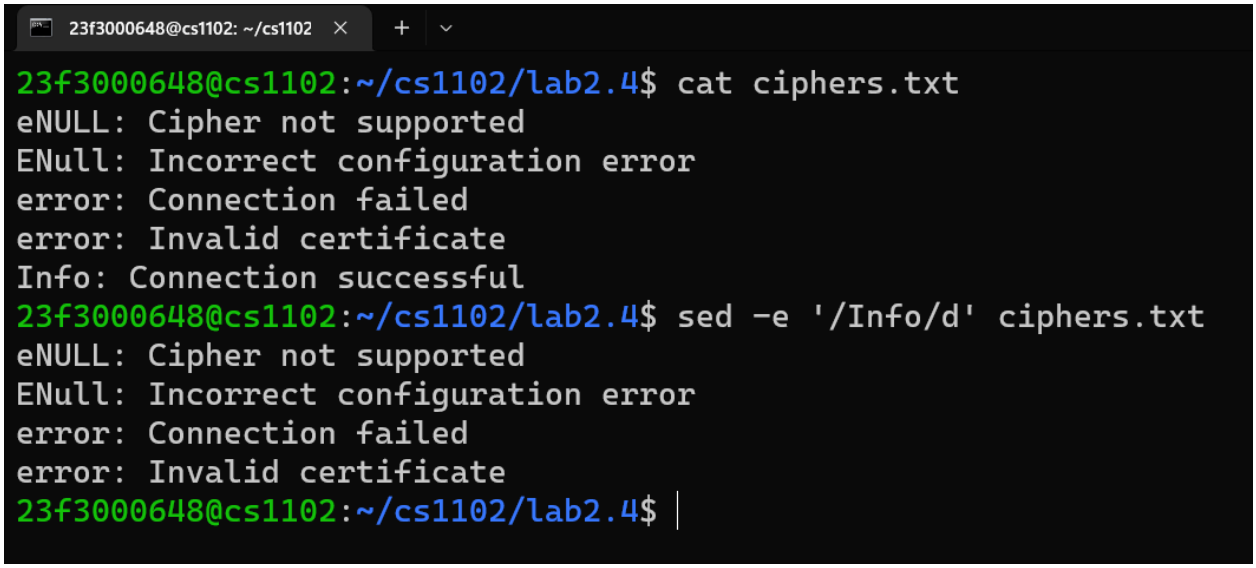


Deleting Lines:

Task: Delete all lines containing 'error'.



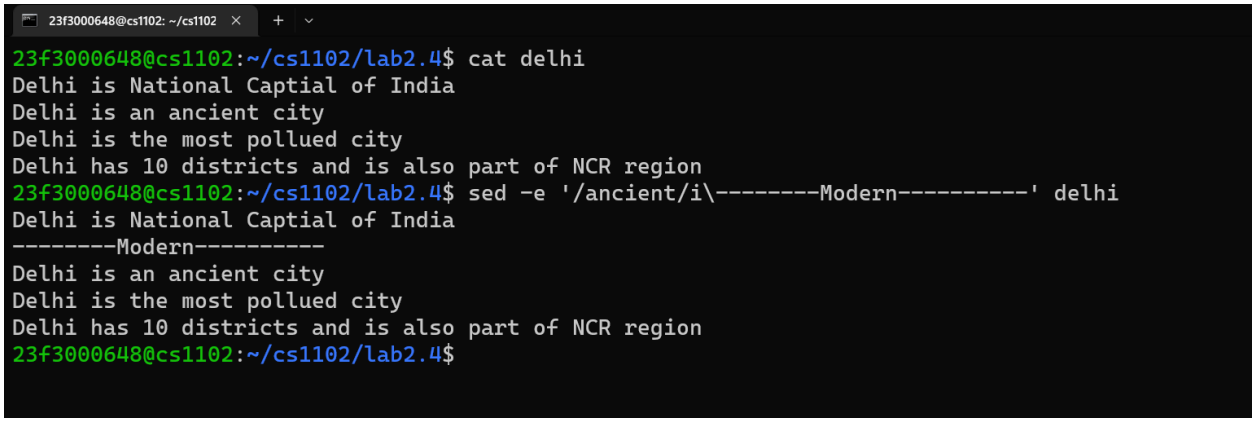
Task: Delete all lines containing 'Info'



Insertion:

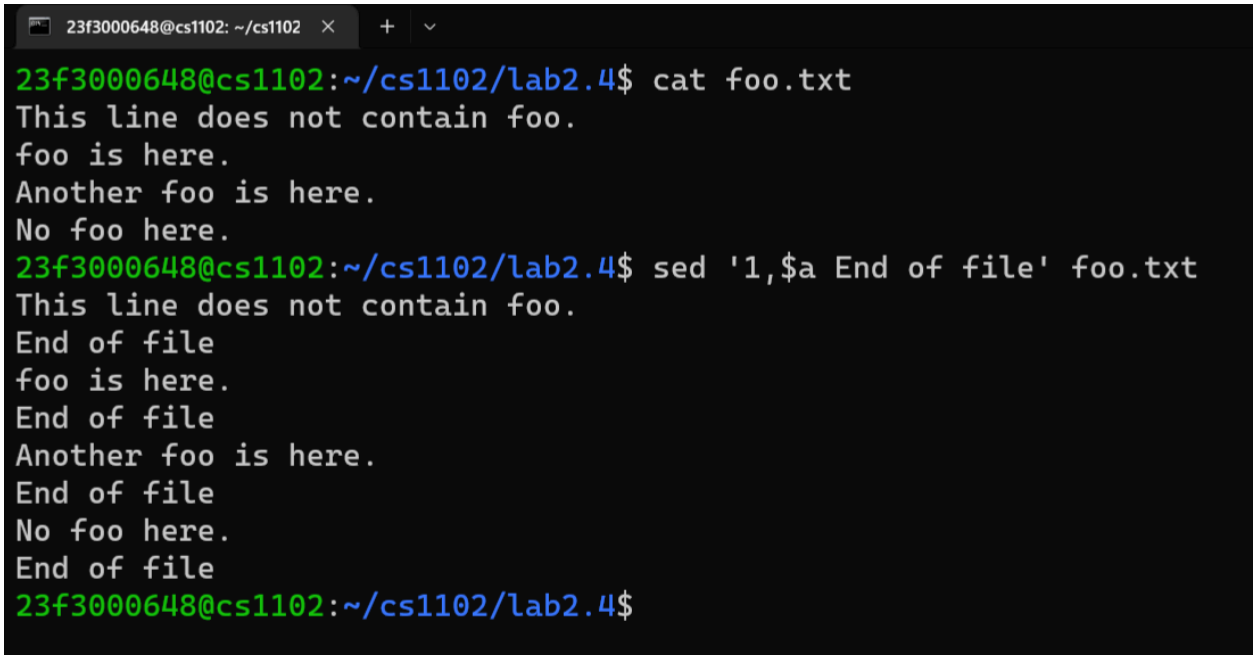
Task: Insert a line 'Hello World' before each line containing 'foo'.

Insert a line '--------Modern—----' before each line containing 'anicent'.



Appending Text:

Task: Append 'End of file' at the end of each line.



Task: Append ‘National Capital Territory’ at the end of each line

```
23f3000648@cs1102: ~/cs1102
23f3000648@cs1102:~/cs1102/lab2.4$
23f3000648@cs1102:~/cs1102/lab2.4$ cat delhi
Delhi is National Captial of India
Delhi is an ancient city
Delhi is the most pollued city
Delhi has 10 districts and is also part of NCR region
23f3000648@cs1102:~/cs1102/lab2.4$ sed '1,$a National Captial Territory' delhi
Delhi is National Captial of India
National Captial Territory
Delhi is an ancient city
National Captial Territory
Delhi is the most pollued city
National Captial Territory
Delhi has 10 districts and is also part of NCR region
National Captial Territory
23f3000648@cs1102:~/cs1102/lab2.4$
```

Deleting Specific Lines:

Task: Delete lines 5 to 10 from a file.

```
23f3000648@cs1102:~/cs1102$ cat linux.txt
Linux is an open-source operating system kernel.
It was created by Linus Torvalds in 1991.
Linux is known for its stability and flexibility.
It is widely used in servers, desktops, and embedded systems.
Popular Linux distributions include Ubuntu, Fedora, and Debian.
Linux supports multi-user and multitasking environments.
The Linux file system hierarchy starts with the root directory "/".
The shell is a command-line interface to interact with the OS.
Bash is one of the most commonly used Linux shells.
Linux uses permissions to manage file access: read, write, execute.
Package managers like apt and yum simplify software installation.
Linux is popular among developers for its robust tools and flexibility.
Open-source contributions continuously improve the Linux ecosystem.
Many IoT devices and smartphones use Linux-based systems like Android.
Learning Linux commands is essential for effective system management.
23f3000648@cs1102:~/cs1102$ sed '5,10d' linux.txt
Linux is an open-source operating system kernel.
It was created by Linus Torvalds in 1991.
Linux is known for its stability and flexibility.
It is widely used in servers, desktops, and embedded systems.
Package managers like apt and yum simplify software installation.
Linux is popular among developers for its robust tools and flexibility.
Open-source contributions continuously improve the Linux ecosystem.
Many IoT devices and smartphones use Linux-based systems like Android.
Learning Linux commands is essential for effective system management.
23f3000648@cs1102:~/cs1102$
```

Delete lines 1 to 14 from a file.

```
23f3000648@cs1102:~/cs1102$ cat linux.txt
Linux is an open-source operating system kernel.
It was created by Linus Torvalds in 1991.
Linux is known for its stability and flexibility.
It is widely used in servers, desktops, and embedded systems.
Popular Linux distributions include Ubuntu, Fedora, and Debian.
Linux supports multi-user and multitasking environments.
The Linux file system hierarchy starts with the root directory "/".
The shell is a command-line interface to interact with the OS.
Bash is one of the most commonly used Linux shells.
Linux uses permissions to manage file access: read, write, execute.
Package managers like apt and yum simplify software installation.
Linux is popular among developers for its robust tools and flexibility.
Open-source contributions continuously improve the Linux ecosystem.
Many IoT devices and smartphones use Linux-based systems like Android.
Learning Linux commands is essential for effective system management.
23f3000648@cs1102:~/cs1102$ sed '1,14d' linux.txt
Learning Linux commands is essential for effective system management.
23f3000648@cs1102:~/cs1102$
```

Replacing Nth Occurrence:

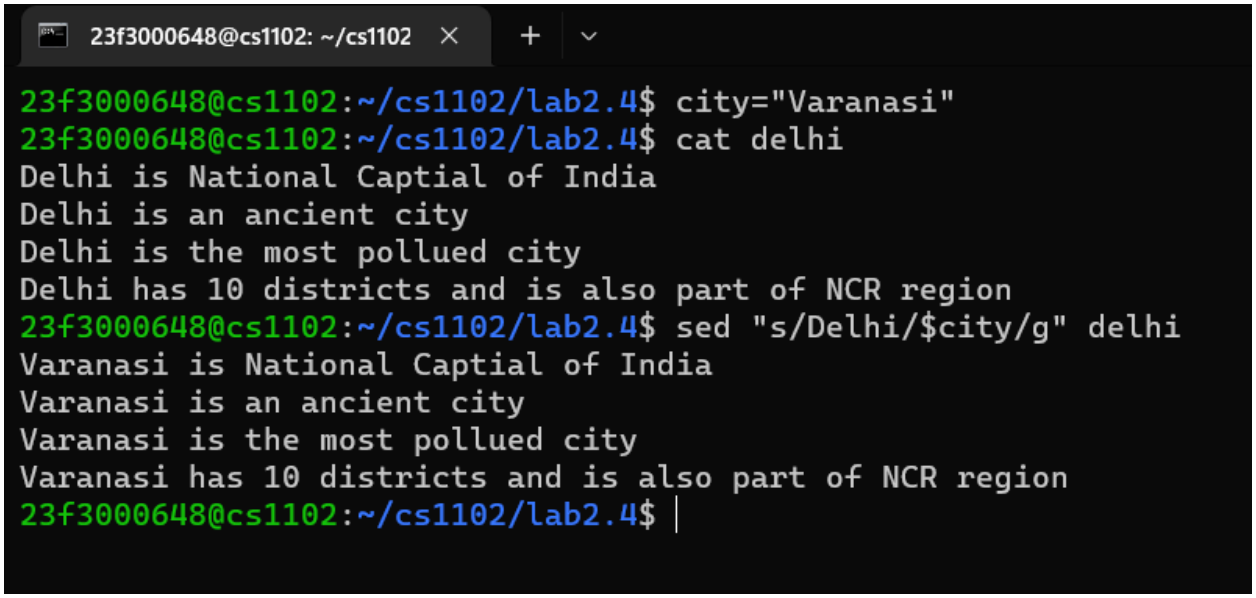
Task: Replace the third occurrence of 'old' with 'new' in each line.

Using Variables in Replacement:

Task: Replace 'apple' with the value of a shell variable 'fruit'.

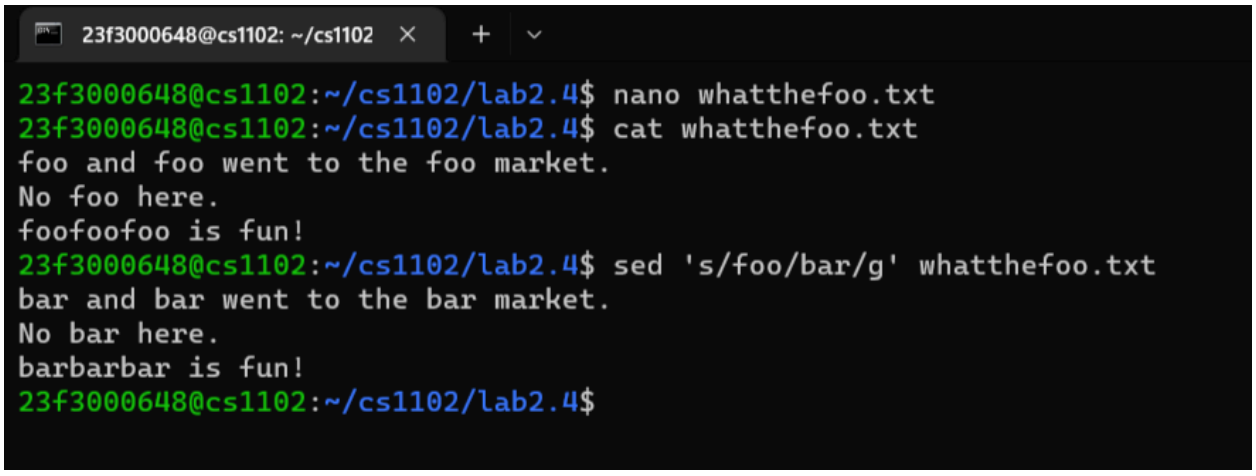
```
23f3000648@cs1102: ~/cs1102
23f3000648@cs1102:~/cs1102/lab2.4$ fruit="orange"
23f3000648@cs1102:~/cs1102/lab2.4$ cat fruits.txt
Fruits are healthy for health
It is always said that an apple a day keeps doctor away so such is the importance of a fruit in our life.
So we should eat apple to reduce our health expenses and to be healthy and to  be away from doctor
.
23f3000648@cs1102:~/cs1102/lab2.4$ sed "s/apple/$fruit/g" fruits.txt
Fruits are healthy for health
It is always said that an orange a day keeps doctor away so such is the importance of a fruit in our life.
So we should eat orange to reduce our health expenses and to be healthy and to  be away from doctor
.
23f3000648@cs1102:~/cs1102/lab2.4$
```

Replace 'Delhi' with the value of a shell variable 'city'.



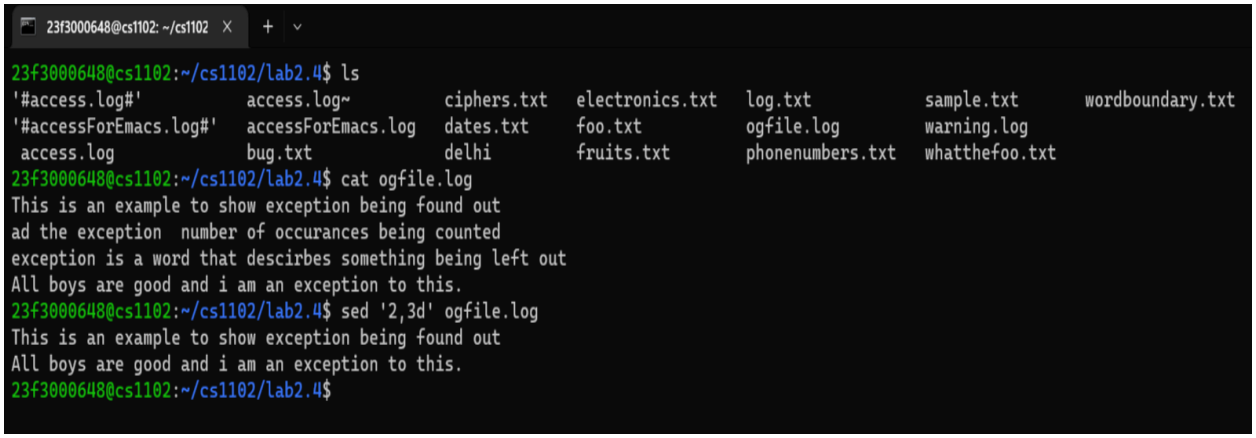
Global Flag (g):

Task: Apply an operation globally (all occurrences) on each line.

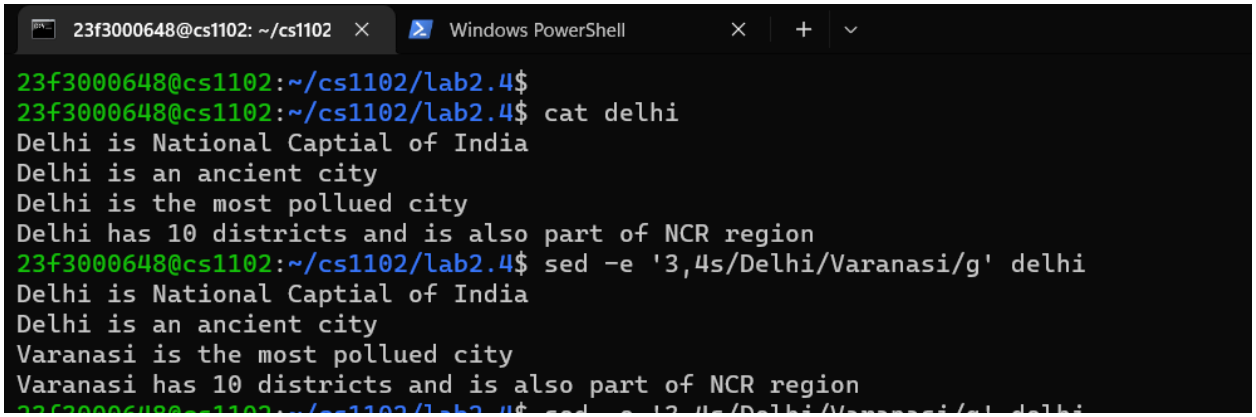


Addressing (1,2):

Task: Specify line numbers or patterns to apply operations on.



Addressing (3,4)



Transformation (y):

Task:Transform characters based on specified mappings. (Examine what y does in sed and report it)

y/source/dest/

Transliterate the characters in the pattern space which appear in source to the corresponding character in dest.

```
23f3000648@cs1102: ~/cs1102
23f3000648@cs1102:~/cs1102/lab2.4$ cat helloworld..txt
hello world
sed is fun
vowels are important
23f3000648@cs1102:~/cs1102/lab2.4$ sed 'y/aeiou/AEIOU' helloworld..txt
sed: -e expression #1, char 13: unterminated `y' command
23f3000648@cs1102:~/cs1102/lab2.4$ sed 'y/aeiou/AEIOU/' helloworld..txt
hEllO wOrld
sEd Is fUn
vOwEls ArE ImpOrtAnt
23f3000648@cs1102:~/cs1102/lab2.4$
```

Example2:

```
23f3000648@cs1102: ~/cs1102
23f3000648@cs1102:~/cs1102/lab2.4$ ls
'#access.log#'           access.log~          ciphers.txt   electronics.txt   helloworld..txt   phonenumbers.txt   whatthefoo.txt
'#accessForEmacs.log#'   accessForEmacs.log   dates.txt     foo.txt           log.txt           sample.txt         wordboundary.txt
 access.log              bug.txt              delhi         fruits.txt        ogfile.log        warning.log
23f3000648@cs1102:~/cs1102/lab2.4$ cat warning.log
This is a warning to shut off the computer.

This Warning should be ignored

This waRning can't be ignored
23f3000648@cs1102:~/cs1102/lab2.4$ sed 'y/Wa/Won/' warning.log
sed: -e expression #1, char 9: strings for `y' command are different lengths
23f3000648@cs1102:~/cs1102/lab2.4$ sed 'y/Wa/Wo/' warning.log
This is o worning to shut off the computer.

This Worning should be ignored

This woRning con't be ignored
23f3000648@cs1102:~/cs1102/lab2.4$
```

Printing Line Numbers (=):

Task: Print line numbers for each line.

```
23f3000648@cs1102: ~/cs1102
23f3000648@cs1102:~/cs1102/lab2.4$ sed "=" warning.log
1
This is a warning to shut off the computer.
2

3
This Warning should be ignored
4

5
This waRning can't be ignored
23f3000648@cs1102:~/cs1102/lab2.4$
```

```
23f3000648@cs1102: ~/cs1102
23f3000648@cs1102:~/cs1102/lab2.4$ ls
'#access.log#'          access.log~          ciphers.txt   electronics.txt   helloworld..txt   phonenumbers.txt   whatthefoo.txt
'#accessForEmacs.log#'  accessForEmacs.log   dates.txt     foo.txt           log.txt           sample.txt         wordboundary.txt
 access.log             bug.txt              delhi         fruits.txt        ogfile.log        warning.log
23f3000648@cs1102:~/cs1102/lab2.4$ cat delhi
Delhi is National Captial of India
Delhi is an ancient city
Delhi is the most pollued city
Delhi has 10 districts and is also part of NCR region
23f3000648@cs1102:~/cs1102/lab2.4$ sed '=' delhi
1
Delhi is National Captial of India
2
Delhi is an ancient city
3
Delhi is the most pollued city
4
Delhi has 10 districts and is also part of NCR region
23f3000648@cs1102:~/cs1102/lab2.4$
```

Suppressing Output (-n):

Task: Suppress automatic printing of pattern space.

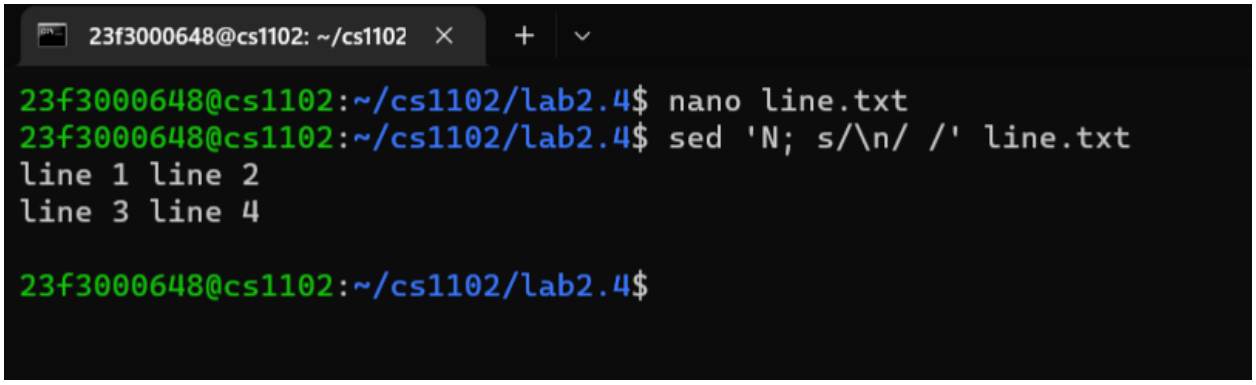
```
23f3000648@cs1102: ~/cs1102
23f3000648@cs1102:~/cs1102/lab2.4$ ls
'#access.log#'          access.log~          ciphers.txt   electronics.txt   helloworld..txt   phonenumbers.txt   whatthefoo.txt
'#accessForEmacs.log#'  accessForEmacs.log   dates.txt     foo.txt           log.txt           sample.txt         wordboundary.txt
 access.log             bug.txt              delhi         fruits.txt        ogfile.log        warning.log
23f3000648@cs1102:~/cs1102/lab2.4$ cat delhi
Delhi is National Captial of India
Delhi is an ancient city
Delhi is the most pollued city
Delhi has 10 districts and is also part of NCR region
23f3000648@cs1102:~/cs1102/lab2.4$ sed -n " " delhi
23f3000648@cs1102:~/cs1102/lab2.4$
```

Another example

```
23f3000648@cs1102:~/cs1102/lab2.4$
23f3000648@cs1102:~/cs1102/lab2.4$ ls
'#access.log#'          access.log~          ciphers.txt   electronics.txt   helloworld..txt   phonenumbers.txt   whatthefoo.txt
'#accessForEmacs.log#'  accessForEmacs.log   dates.txt     foo.txt           log.txt           sample.txt         wordboundary.txt
 access.log             bug.txt              delhi         fruits.txt        ogfile.log        warning.log
23f3000648@cs1102:~/cs1102/lab2.4$ cat ogfile.log
This is an example to show exception being found out
ad the exception  number of occurances being counted
exception is a word that descirbes something being left out
All boys are good and i am an exception to this.
23f3000648@cs1102:~/cs1102/lab2.4$ sed -n '/boys/p' ogfile.log
All boys are good and i am an exception to this.
23f3000648@cs1102:~/cs1102/lab2.4$
```

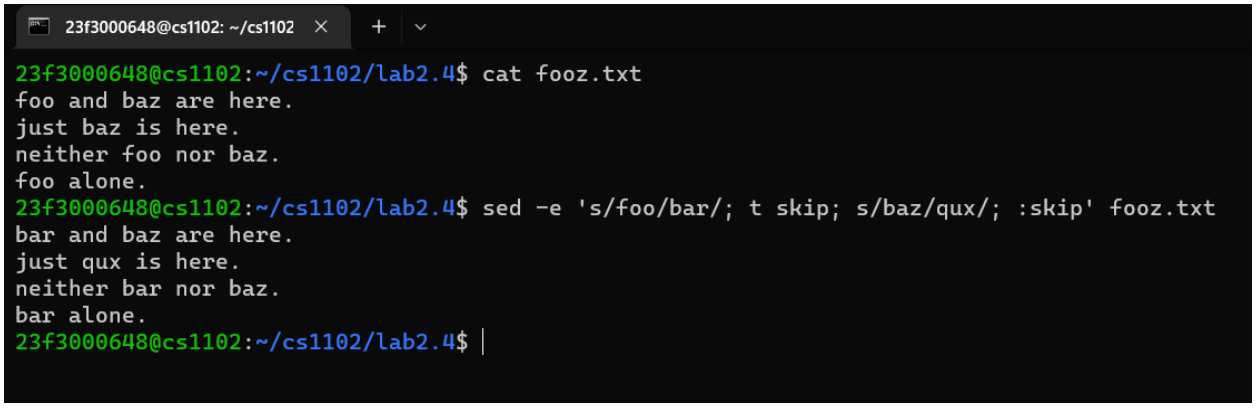
Appending Next Line (N):

Task: Append the next line to the pattern space.



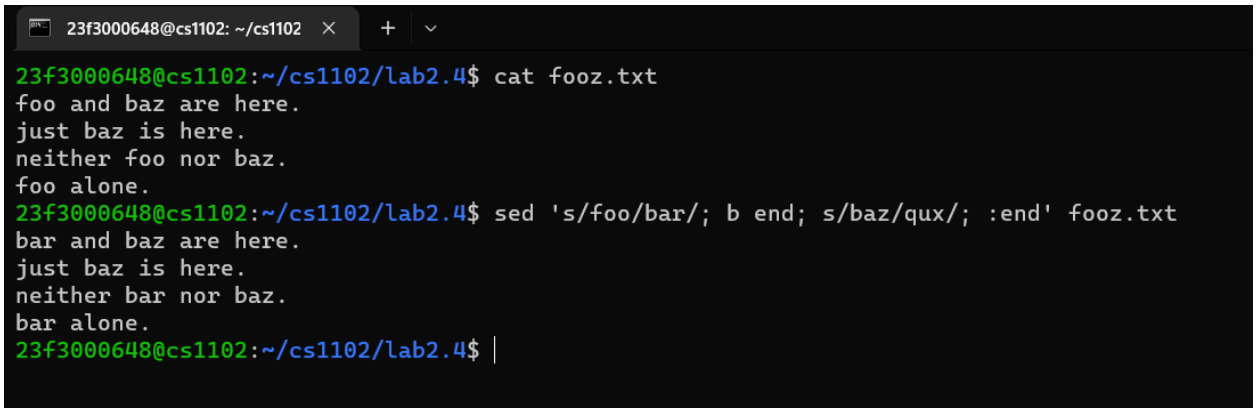
Conditional Branching (b): (Bonus)

Task: Use branches to a specified label if the previous substitution was successful.



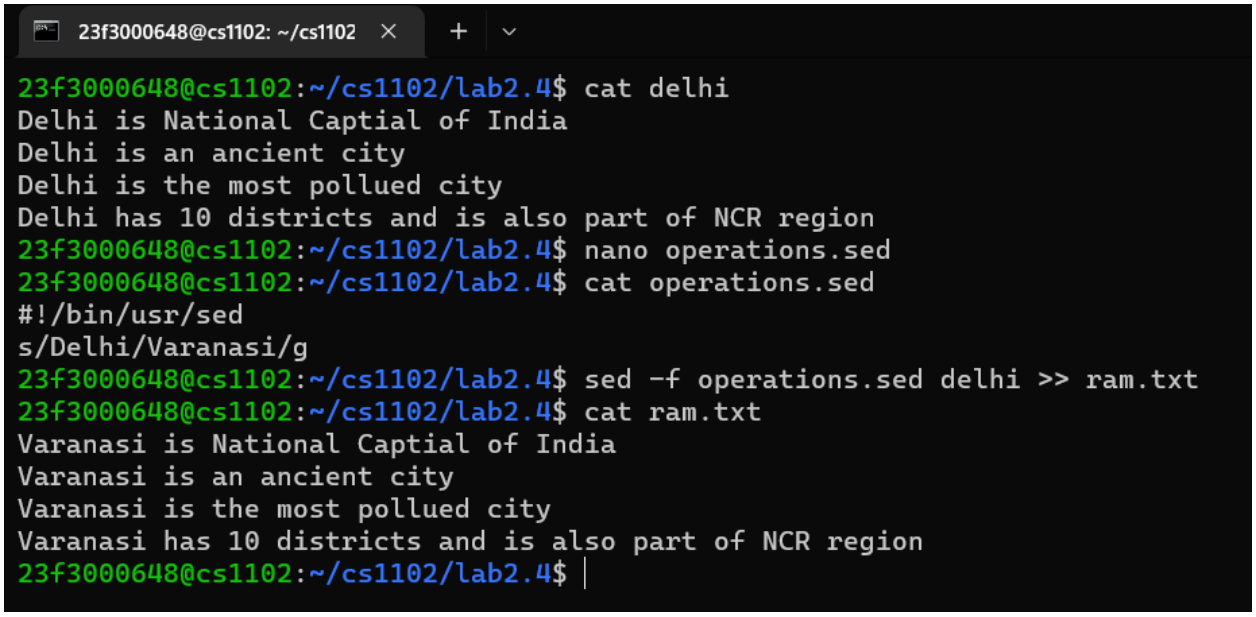
Branching to Label (b): (Bonus)

Task: Use Branches to a specified label unconditionally.



Reading External File (r): (Bonus)

Task: Read content from an external file and appends it to the output.


23f3000648@cs1102: ~/cs1102
23f3000648@cs1102:~/cs1102/lab2.4$ cat delhi
Delhi is National Captial of India
Delhi is an ancient city
Delhi is the most pollued city
Delhi has 10 districts and is also part of NCR region
23f3000648@cs1102:~/cs1102/lab2.4$ nano operations.sed
23f3000648@cs1102:~/cs1102/lab2.4$ cat operations.sed
#!/bin/usr/sed
s/Delhi/Varanasi/g
23f3000648@cs1102:~/cs1102/lab2.4$ sed -f operations.sed delhi >> ram.txt
23f3000648@cs1102:~/cs1102/lab2.4$ cat ram.txt
Varanasi is National Captial of India
Varanasi is an ancient city
Varanasi is the most pollued city
Varanasi has 10 districts and is also part of NCR region
23f3000648@cs1102:~/cs1102/lab2.4$

Linux Lab 5 activities

Activity 1:

1. Add:

```
#!/bin/bash

# Add the values of lines in a file
# Usage: add <file>
# or command | add
#
# Contributed by: TA Sayan Ghosh

paste -s -d '+' "$@" | bc
```

Breakdown:

1. Shebang (`#!/bin/bash`):
The script starts with the shebang (`#!/bin/bash`), which tells the system that this script should be run using the Bash shell.

2. Comments (#):
Lines starting with # are comments. They are for documentation purposes only.

The first comment explains the script's purpose: "Add the values of lines in a file."

The second comment shows how to use the script: `add <file>` or `command | add`.

The final comment indicates the contributor's name.

3.`Paste` Command:
`paste` is used to merge lines of files. The `paste` command is used here with the following options:

`-s` (serial): This option makes `paste` combine all lines from the input into a single line.

`-d '+'`: This option sets the delimiter between the lines to a plus sign (+), so the lines from the file or the output of the command are joined with +.

4.`"$@"`:
This is a special variable in Bash that represents all the arguments passed to the script. In this case, it's the file or the command output provided to the script.
For instance, if you run `add numbers.txt`, `$@` will expand to `numbers.txt`.

5.`| bc`:
The pipe (`|`) sends the output of the `paste` command to the `bc` command. `bc` is a calculator utility in Unix-like systems that can evaluate mathematical expressions.
For example, if `paste` outputs 1+2+3, the pipe (`|`) sends this to `bc`, which calculates and returns the result `6`.

Pseudo Code:

1. Start script execution.

2. Read the file(s) or command output passed as arguments.

3. Join all the lines of the file (or command output) into a single string with "+" as the separator.

   - Example: Convert "1", "2", "3" to "1+2+3".

4. Send the resulting string (e.g., "1+2+3") to the `bc` command for evaluation.

5. Output the result of the calculation (e.g., 6).

6. End script execution.

New Commands:

There are three main commands in the script that we need to explore further:

1. `paste`:

The `paste` command is used to merge lines from files.
Use Cases:

- Merge columns: Combine multiple columns from files into a single line. E.g., `paste file1 file2` merges the columns of `file1` and `file2`.
- Join lines with a custom delimiter: Join lines using a specific delimiter, e.g., `paste -d ',' file1 file2` joins lines from `file1` and `file2` with a comma.
- Create a summary file: Combine various parts of a dataset (e.g., two different reports) to create a summary file with all information in a single line.

2. `"$@"`:

This is a special variable in Bash that represents all the arguments passed to the script.

Use Cases:

- Iterating over arguments: Use `"$@"` in a `for` loop to iterate over all arguments passed to a script.

Passing arguments to another script: Forward arguments from one script to another using `"$@"`.

Handling multiple files in a script: Process multiple files passed as arguments in a script.

3. bc:

bc is a command-line calculator utility for performing mathematical operations.

Use Cases:

Simple arithmetic: Perform basic operations like addition, subtraction, multiplication, and division. E.g., `echo "5+3" | bc` outputs 8.

Precision control: Perform operations with floating-point precision. E.g., `echo "scale=2; 22/7" | bc` outputs `3.14`.

Complex calculations: Use bc for advanced mathematical operations like square roots, exponentiation, etc. E.g., `echo "sqrt(9)" | bc` outputs 3.

2. Batsat:

```
#!/bin/bash
#
# Print Battery statistics on terminal
# Contributed by: TA Prabuddh Mathur
#
upower -i "$(upower -e | grep BAT)" | grep --color=never -E "state|to full|to empty|percentage"
```

Breakdown:

1. Shebang (`#!/bin/bash`):
This is the shebang line, which indicates that the script should be run using the Bash shell.

2. Comments (#):
The comments provide information about the script:

> First comment: Describes the script's purpose ("Print Battery statistics on terminal").
>
> Second comment: Indicates the contributor's name.

`3.upower -e`:
The `upower` command is used to interact with power-related statistics (e.g., battery, AC adapter, etc.). The `-e` option lists the available power devices (e.g., battery, AC, etc.).

> The `upower -e` command outputs something like `/org/freedesktop/UPower/devices/battery_BAT0`, which represents the path to the battery device.

`4.grep BAT`:

The `grep` command searches for lines containing the string `BAT` from the output of `upower -e`. This is used to filter out the path to the battery device. If there are multiple devices (e.g., an external battery and internal battery), this filters out the one that contains `BAT`, which is typically associated with battery devices.

5.upower -i "$(upower -e | grep BAT)":

This command takes the path to the battery device (provided by the previous `grep BAT` command) and passes it as an argument to `upower -i`.

The `-i` option with `upower` provides detailed information about the battery device (e.g., status, charge level, time to full, etc.).

6.grep --color=never -E "state|to full|to empty|percentage":

This part of the command pipes the output from `upower -i` to `grep`, which filters for specific patterns:

> `--color=never`: Disables colored output, which is typically used for highlighting matches.
>
> `-E`: Enables extended regular expressions, allowing the use of more complex patterns.
>
> `"state|to full|to empty|percentage"`: This pattern looks for lines containing "state", "to full", "to empty", or "percentage". These are the key battery statistics the user is interested in.

Pseudo Code:

1. Start script execution.
2. Run the `upower -e` command to list all power devices.
3. Filter the output to find the battery device path using `grep BAT`.

4. Run `upower -i` with the battery device path to get detailed battery information.

5. Filter the output to show only the relevant statistics (state, time to full, time to empty, and percentage).

6. Display the filtered information on the terminal.

7. End script execution.

New Commands:

1. `upower`:

   `upower` is a command-line utility that interacts with power devices (battery, AC, etc.). It provides battery status, power device properties, and energy consumption statistics.

Use Cases of `upower`:

Battery status: Use `upower -i /org/freedesktop/UPower/devices/battery_BAT0` to check battery status, including charge percentage and time remaining.

Monitor energy consumption: Use `upower -d` to display detailed information about the energy consumption of various power devices.

List power devices: Use `upower -e` to list all the power-related devices on the system (batteries, AC adapters, etc.).

2. `Grep`:

`grep` is a command-line utility that searches for patterns in text. In this script, `grep` is used to filter the output of commands based on specific patterns.

Use Cases of `grep`:

Search for specific text in files: Use `grep` to find lines in a file containing a specific word or phrase. E.g., `grep "error" logfile.txt`.

Filter command output: Use `grep` to filter output from other commands, such as filtering logs for specific events. E.g., `dmesg | grep "usb"`.

Search recursively in directories: Use `grep -r` to search for a pattern in all files within a directory and its subdirectories. E.g., `grep -r "function_name" ./src`.

`--color=never` Option for `grep`:

This option disables colored highlighting in `grep` output, ensuring the output is plain text without any color formatting.

Use Cases of `--color=never`:

Disable color in scripts: Use this option in scripts to avoid colored output, which might cause issues when redirecting the output to files or piping it to other commands.

Search in environments without color support: In terminal environments that do not support color, this option disables the color highlighting.

Improve readability in logs: Use it when searching through log files that are often processed by other tools, making it easier to parse plain output.

3. Add_user:

```
#!/bin/bash
#
# Add a user with a password, homedir, and bash shell
# Contributed by: TA Gaurav Batheja
#
username="$1"
password="$2"
useradd "$username" -m -c /bin/bash
echo "$username:$password" | chpasswd
```

Breakdown:

1. Shebang line (`#!/bin/bash`):

Purpose: Indicates the script should be executed using the `bash` shell. It tells the operating system which interpreter to use to run the script.

2. Comment Block (`# Add a user with a password, homedir, and bash shell`):

Purpose: Describes the purpose of the script. This is for documentation and does not affect the execution of the script.

3. Variable Assignment:

`username="$1"`

Purpose: Captures the first argument passed to the script and assigns it to the variable `username`.

`password="$2"`

Purpose: Captures the second argument passed to the script and assigns it to the variable `password`.

4. useradd "$username" -m -c /bin/bash:

Purpose: This command is intended to create a new user with the specified username.

Incorrect Option Usage:

`-m` creates a home directory for the user.

`-c` is used for adding a comment (typically the user's full name), but the script mistakenly uses `/bin/bash` here, which would cause the script to fail. The correct option for setting the shell is `-s /bin/bash`.

5.echo "$username:$password" | chpasswd:

Purpose: This sets the password for the user by passing the username and password to the `chpasswd` command. The command reads the input in the form `username:password` and updates the password.

Pseudo Code:

1. Start Script Execution:

   Begin execution by calling the script with arguments (username, password).

2. Capture Input Arguments:

    Store the first argument passed to the script as the `username`.

    Store the second argument passed to the script as the `password`.

Create a New User:

Run the `useradd` command to create the user:

`-m` option to create a home directory.

`-s /bin/bash` option to set the default shell to bash.

`-c` option (which should contain the user's full name, but was mistakenly used for the shell in the original script).

Set User Password:

Run the `echo` command to pass the username and password to `chpasswd`.

`chpasswd` will update the password for the created user.

End Script Execution:

Once the user and password are set, the script ends.

New Commands:

1. `useradd` Command:

    Purpose**:** Adds a new user to the system.

Use Cases:

a. Create a new user with a home directory:

useradd -m username

b. Set a specific shell for a new user:

useradd -m -s /bin/bash username

c. Create a user and specify a group:

useradd -m -g groupname username

4. Check_no_logged_in:

```
#!/bin/bash
#
# Print count of logged in users
# Contributed by: TA Bathula Sharath Kumar
#
users=$(who | wc -l)
echo "Number of currently logged-in users: $users"
```

Breakdown:

1. Shebang (`#!/bin/bash`):

This is called the *shebang* or *hashbang*. It tells the operating system that the script should be executed using the Bash shell (`/bin/bash`).

2. Comment Block:

The lines starting with # are comments. Comments are used to explain the code and are not executed. In this case:

`# Print count of logged in users`: This explains what the script does.

`# Contributed by: TA Bathula Sharath Kumar`: A contributor's note.

3. Variable Assignment (`users=$(who | wc -l)`):

`users=$(...)` stores the result of the command inside the parentheses into the variable `users`.

`who`: This command shows a list of currently logged-in users on the system.

`|`: The pipe operator takes the output of the command on the left (`who`) and sends it as input to the command on the right (`wc -l`).

`wc -l`: This command counts the number of lines in the input it receives. Since `who` outputs one line per user, `wc -l` effectively counts the number of logged-in users.

The result is stored in the `users` variable.

4. Output (`echo "Number of currently logged-in users: $users"`):

`echo` is used to print the result to the terminal.

`"$users"` will expand to the value of the `users` variable (i.e., the count of logged-in users).

Pseudo Code:

1. Start

2. Initialize a variable `users`:

    Run the `who` command to get a list of logged-in users.

    Pipe the output of `who` into `wc -l` to count the number of lines.

    Store the result (the number of logged-in users) into the variable `users`.

3. Print the result:

    Use `echo` to output a message that shows the number of currently logged-in users, by referencing the `users` variable.

4. END

New Commands:

1. `who` Command:

    The `who` command displays who is logged into the system.

Use cases:

    View current logged-in users:
    Who

    Check specific user's login information

    Who username

    List logged-in users with terminal information

Who -u

2. `wc` Command:

   The wc command stands for "word count" but it can be used to count lines, words, and characters in a file or input stream.

   Use cases:

   1. Count the number of lines in a file

      Wc -l filename

   2. Count world in a file

      Wc -w filename

   3. Count characters in a file

      Wc -c filename

5. Check_size:

```bash
#!/bin/bash
# Contributed By: TA Ashwin Hebbar (21f1003155)
# Function to check size of a file or a directory
check_size() {
    if [ -e "$1" ]; then
        du -sh "$1"
    else
        echo "File or directory not found."
    fi
}

check_size "$1"
```

Breakdown:

1. Shebang (`#!/bin/bash`):

This is called the *shebang* or *hashbang*. It tells the operating system that the script should be executed using the Bash shell (`/bin/bash`).

2. Comment Block:

The lines starting with # are comments. Comments are used to explain the code and are not executed.

`# Contributed By: TA Ashwin Hebbar (21f1003155)`: A contributor's note.

`# Function to check size of a file or a directory`: A description of what the script does.

3. Function Definition (`check_size()`):

The script defines a function named `check_size`. Functions allow you to encapsulate a block of code and call it multiple times. In this case, the function checks the size of a given file or directory.

`check_size()`: The function that will be used to check the size.

Input: The function takes one parameter, which is expected to be the path to a file or directory.

Output: It either prints the size of the file/directory or an error message if the file/directory does not exist.

4. If Statement (`if [ -e "$1" ]; then`):

`if [ -e "$1" ]`: This checks if the argument (`$1`) passed to the script exists.

`-e` checks if a file or directory exists (whether it's a file, directory, or other types of filesystem objects).

If the condition is true (i.e., the file or directory exists), the script proceeds to execute the `du -sh "$1"` command.

Else: If the condition is false (i.e., the file or directory doesn't exist), it will print `"File or directory not found."`.

5. Disk Usage Command (`du -sh "$1"`):

   `du`: The `du` command is used to estimate file and directory space usage.

   `-s`: This option summarizes the total size of the file or directory (doesn't list individual files).

   `-h`: This makes the output human-readable, i.e., it displays the size in a format like KB, MB, GB instead of just bytes.

   $1: This is the argument passed to the function, which could be a file or directory path.

6. Function Call (`check_size "$1"`):

   This calls the `check_size` function and passes the first command-line argument ($1) to it.

   $1 refers to the first argument given when executing the script. This argument will be the file or directory path whose size we want to check.

Pseudo Code:

1. Start.

2. Define a function `check_size`:

    Input: Accept a file or directory path as the argument.

    Check if the path exists:

        If the path exists, run the `du -sh` command to check the size and print it.

        If the path does not exist, print "`File or directory not found.`".

3. Call the function `check_size`, passing the first argument of the script ($1), which is the file or directory path.
4. End.

New Commands:

1. du Command:

    The du (disk usage) command estimates and reports the disk space used by files and directories.

    Use cases:

    1. Check the size of a directory:

        du -sh /path/to/directory

2. Check the size of a specific file:

    du -sh /path/to/file

3. Display the disk usage of all subdirectories:

```
du -sh /path/to/directory/*
```

2. -e Option (File Existence Check):

The -e option is used with [(the test command) to check if a file or directory exists.

Use cases:

Check if a file exists:

```
if [ -e /path/to/file ]; then echo "File exists"; fi
```

Check if a directory exists:

```
if [ -e /path/to/directory ]; then echo "Directory exists"; fi
```

Check if a symbolic link exists:

```
if [ -e /path/to/symlink ]; then echo "Symlink exists"; fi
```

Activity 2:

1. File Management Script:

Pseudo code:

```
#!/bin/bash

# Check if a filename was provided as an argument
if [ $# -ne 1 ]; then
  echo "Usage: $0 <filename>"
  exit 1
```

```
fi

filename="$1"


# Check if the file exists
if [ -e "$filename" ]; then
  echo "File '$filename' exists."


   # Check if the file is readable
  if [ -r "$filename" ]; then
    echo "File '$filename' is readable."
  else
    echo "File '$filename' is not readable."
  fi

  # Check if the file is writable
  if [ -w "$filename" ]; then
    echo "File '$filename' is writable."
  else
    echo "File '$filename' is not writable."
  fi


  # Check if the file is executable
  if [ -x "$filename" ]; then
    echo "File '$filename' is executable."
  else
    echo "File '$filename' is not executable."
  fi

else
```

```
  echo "File '$filename' does not exist."
fi
```

Explanation:

1. Shebang Line (`#!/bin/bash`):

Specifies that the script should be executed with the Bash shell.

2. Check for Arguments:

`if [ $# -ne 1 ]; then`: Checks if exactly one argument is provided. If not, it prints usage instructions and exits with a status of 1.

3. Filename Variable:

`filename="$1"`: Assigns the first argument to the `filename` variable.

4. File Existence Check:

`if [ -e "$filename" ]; then`: Checks if the file exists. If it does, it proceeds to check permissions.

5. File Permissions Checks:

`if [ -r "$filename" ]; then`: Checks if the file is readable.

`if [ -w "$filename" ]; then`: Checks if the file is writable.

`if [ -x "$filename" ]; then`: Checks if the file is executable.

6. Output Messages:

Prints messages based on the file's existence and permissions.

7. File Does Not Exist:

If the file does not exist, it prints a message indicating that.

2. Data Processing Script:

Ascript that reads data from a CSV file, performs some calculations or manipulations on the data, and writes the result to another file

Pseudo code:

```
#!/bin/bash

# Check if the correct number of arguments is provided
if [ $# -ne 2 ]; then
  echo "Usage: $0 <input_csv> <output_file>"
  exit 1
fi

input_file="$1"
output_file="$2"

# Check if the input file exists
if [ ! -f "$input_file" ]; then
  echo "Input file '$input_file' does not exist."
  exit 1
fi

# Check if the input file is a CSV
if ! file "$input_file" | grep -q "CSV"; then
  echo "Input file '$input_file' is not a CSV file."
```

```
  exit 1
fi

# Initialize the sum variable
sum=0

# Read the CSV file (skipping the header) and calculate the sum of the values
while IFS=, read -r name value; do
  # Skip the header line
  if [ "$name" != "name" ]; then
    sum=$(echo "$sum + $value" | bc)
  fi
done < "$input_file"

# Write the result to the output file
echo "Total Sum: $sum" > "$output_file"

# Print a success message
echo "Data processing complete. The result has been written to '$output_file'."
```

Explanation:

1.Shebang Line (`#!/bin/bash`):

Indicates that the script should be run with the Bash shell.

2.Argument Check:

`if [ $# -ne 2 ]; then`: Ensures exactly two arguments are provided (input CSV file and output file).

3.File Existence Check:

`if [ ! -f "$input_file" ]; then`: Checks if the input file exists.

4.CSV File Check:

`if ! file "$input_file" | grep -q "CSV"; then`: Ensures the input file is recognized as a CSV file.

5. Initialize Sum Variable:

`sum=0`: Initializes a variable to hold the sum of values.

6. Read and Process CSV:

`while IFS=, read -r name value; do`: Reads the CSV file line by line, using a comma as the delimiter.

`if [ "$name" != "name" ]; then`: Skips the header line.

`sum=$(echo "$sum + $value" | bc)`: Adds each value to the sum using `bc` for arithmetic.

7. Write Result to Output File:

`echo "Total Sum: $sum" > "$output_file"`: Writes the result to the specified output file.

8. Completion Message:

Prints a message indicating that the processing is complete and the result is saved.

3. String Manipulation Script:

A Script that takes a string as input and performs various string manipulation operations such as substring extraction, search and replace, and string concatenation

Pseudo code:

```
#!/bin/bash

# Function to extract a substring from a given string
extract_substring() {
  local str="$1"
  local start="$2"
  local length="$3"
  echo "${str:start:length}"
}

# Function to search and replace a substring in a given string
search_and_replace() {
  local str="$1"
  local search="$2"
  local replace="$3"
  echo "${str//${search}/${replace}}"
}

# Function to concatenate two strings
concatenate_strings() {
  local str1="$1"
  local str2="$2"
  echo "${str1}${str2}"
}

# Check if the correct number of arguments is provided
```

```
if [ $# -lt 3 ]; then
  echo "Usage: $0 <operation> <string1> [string2] [start length] [search
replace]"
  echo "Operations:"
  echo "  extract_substring - Extract a substring from string1"
  echo "  search_and_replace - Search and replace substring in
string1"
  echo "  concatenate_strings - Concatenate string1 and string2"
  exit 1
fi

operation="$1"
string1="$2"
string2="$3"

case "$operation" in
  extract_substring)
    if [ $# -ne 5 ]; then
      echo "Usage: $0 extract_substring <string> <start> <length>"
      exit 1
    fi
    start="$3"
    length="$4"
    extract_substring "$string1" "$start" "$length"
    ;;

  search_and_replace)
    if [ $# -ne 5 ]; then
      echo "Usage: $0 search_and_replace <string> <search>
<replace>"
      exit 1
    fi
```

```
    search="$3"
    replace="$4"
    search_and_replace "$string1" "$search" "$replace"
    ;;

  concatenate_strings)
    if [ $# -ne 4 ]; then
      echo "Usage: $0 concatenate_strings <string1> <string2>"
      exit 1
    fi
    concatenate_strings "$string1" "$string2"
    ;;

  *)
    echo "Invalid operation. Available operations are: extract_substring,
search_and_replace, concatenate_strings."
    exit 1
    ;;
esac
```

Explanation:

1. Shebang Line (`#!/bin/bash`):

Indicates that the script should be run with the Bash shell.

2. Functions:

`extract_substring`:

Takes a string, a start index, and a length to extract a substring.

`${str:start:length}` extracts the substring from the start index with a length of `length`.

`search_and_replace`:

Takes a string, a search substring, and a replace substring.

`${str//${search}/${replace}}` performs the replacement operation.

`concatenate_strings`:

Takes two strings and concatenates them using `${str1}${str2}`.

3. Argument Check:

Ensures the script receives the correct number of arguments for each operation.

Provides usage instructions if the arguments are not as expected.

4.Operation Handling:

Based on the operation specified (`extract_substring`, `search_and_replace`, `concatenate_strings`), the script calls the corresponding function.

5.Usage Instructions:

Displays how to use the script and the available operations if incorrect arguments are provided or if the operation is invalid