Cloud Computing Lab

Submitted To:

Sir Shoaib & Sir Waqas

Submitted By:

Hamail Fatima

Roll No:

2023-BSE-023

Section:

5(A)

Lab # 06

Task 1 – Switch to root with su - and back to a normal user

Goal: Demonstrate switching to the root account using su - and exiting back to your normal user.

1. Set a root password (Ubuntu root is disabled by default; this enables su - temporarily for the lab):

sudo passwd root

Enter a temporary root password for the lab

- Save screenshot as: task1_set_root_password.png

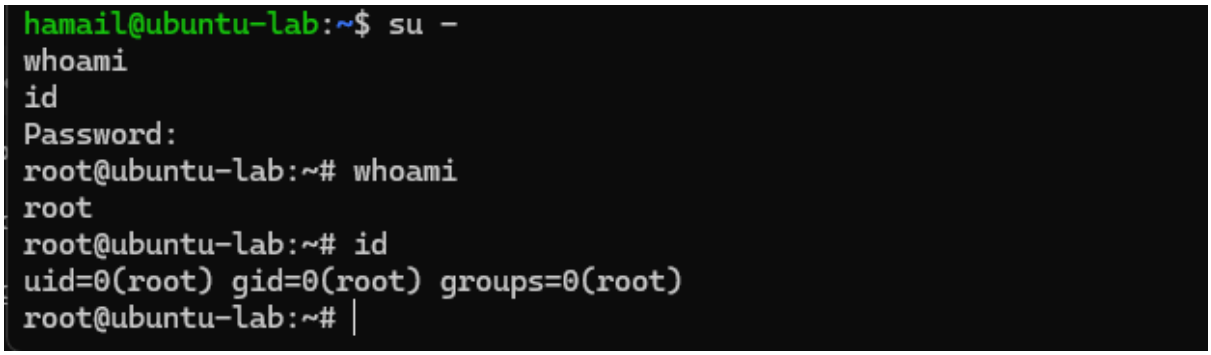
2. Switch to root and verify:

su -

whoami

id

- Save screenshot as: task1_su_root.png



3. Switch back to your normal user:

exit

whoami

- Save screenshot as: task1_exit_to_user.png

```
hamail@ubuntu-lab:~$ su -
whoami
id
Password:
root@ubuntu-lab:~# whoami
root
root@ubuntu-lab:~# id
uid=0(root) gid=0(root) groups=0(root)
root@ubuntu-lab:~# exit
whoami
logout
hamail
uid=1000(hamail) gid=1000(hamail) groups=1000(hamail),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),101(lxd)
hamail@ubuntu-lab:~$
```

Task 2 – Create user tom and verify in passwd/group/shadow

Goal: Create a user named tom, then verify the account in system files.

1. Create user tom (interactive, sets password and home directory):

sudo adduser tom

- Save screenshot as: task2_adduser_tom.png

```
hamail@ubuntu-lab:~$ sudo adduser tom
info: Adding user `tom' ...
info: Selecting UID/GID from range 1000 to 59999 ...
info: Adding new group `tom' (1001) ...
info: Adding new user `tom' (1001) with group `tom (1001)' ...
info: Creating home directory `/home/tom' ...
info: Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for tom
Enter the new value, or press ENTER for the default
        Full Name []: tom
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
info: Adding new user `tom' to supplemental / extra groups `users' ...
info: Adding user `tom' to group `users' ...
hamail@ubuntu-lab:~$
```

2. Verify tom in system files (view and visually confirm presence):

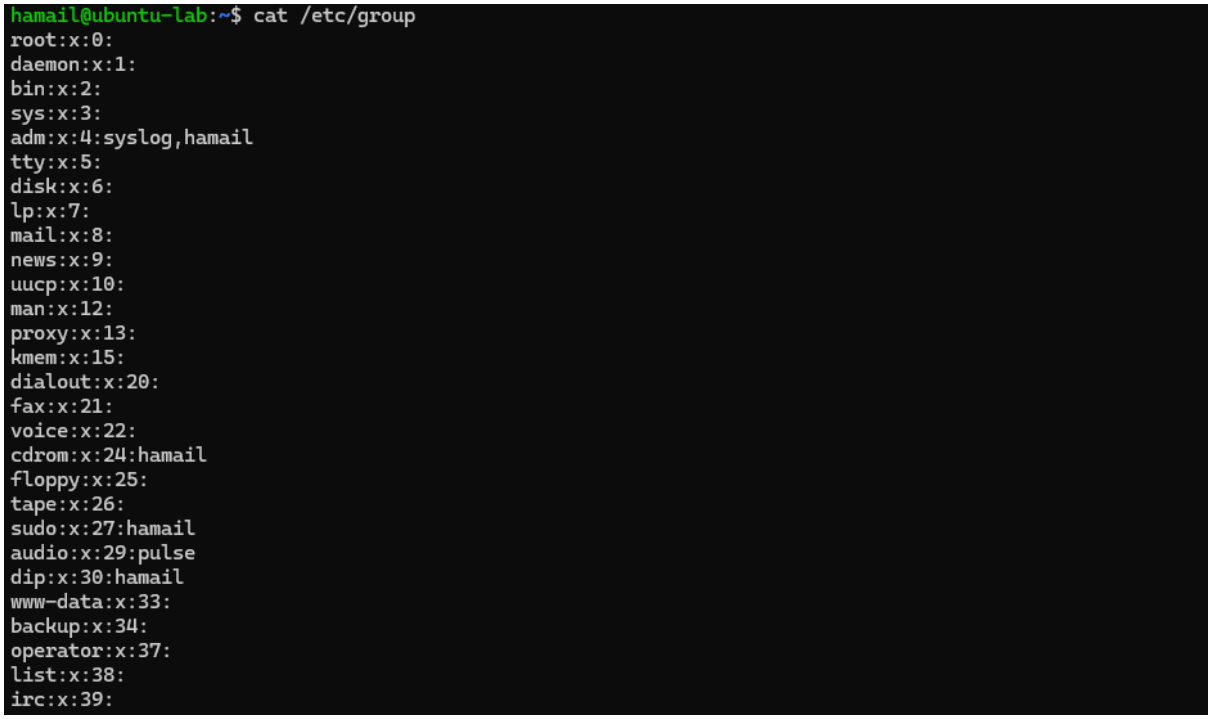
cat /etc/passwd

- Save screenshot as: task2_verify_passwd.png

```
hamail@ubuntu-lab:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
_apt:x:42:65534::/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:998:998:systemd Network Management:/:/usr/sbin/nologin
systemd-timesync:x:997:997:systemd Time Synchronization:/:/usr/sbin/nologin
dhcpcd:x:100:65534:DHCP Client Daemon,,,:/usr/lib/dhcpcd:/bin/false
messagebus:x:101:102::/nonexistent:/usr/sbin/nologin
systemd-resolve:x:992:992:systemd Resolver:/:/usr/sbin/nologin
pollinate:x:102:1::/var/cache/pollinate:/bin/false
polkitd:x:991:991:User for polkitd:/:/usr/sbin/nologin
syslog:x:103:104::/nonexistent:/usr/sbin/nologin
uuidd:x:104:105::/run/uuidd:/usr/sbin/nologin
```

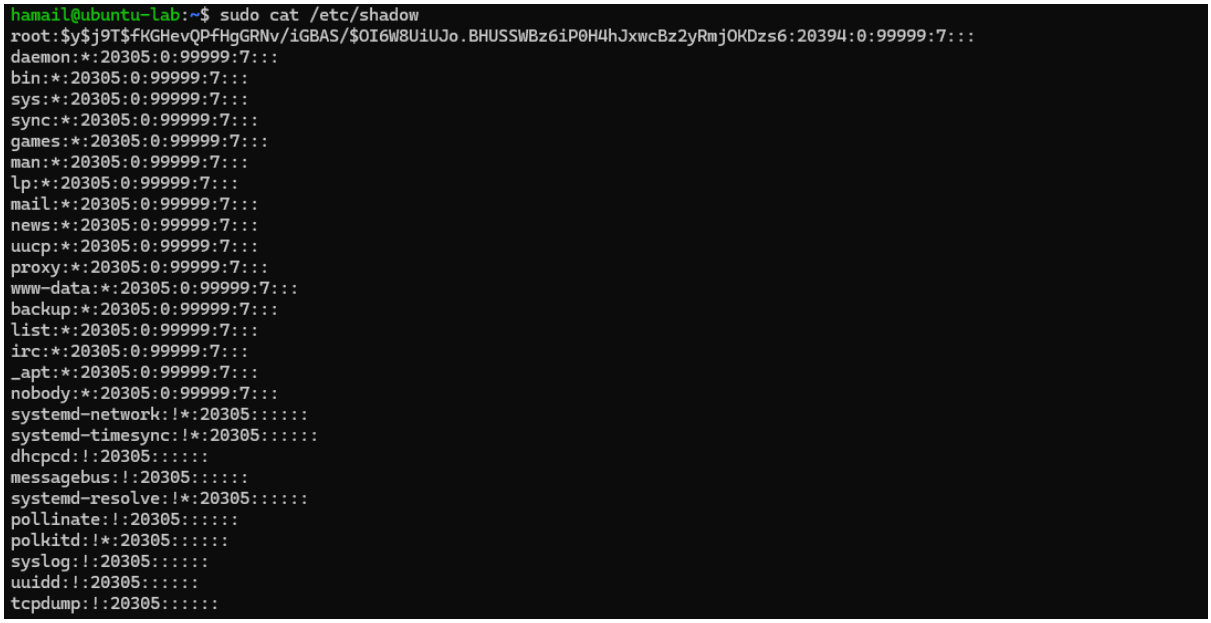
cat /etc/group

- Save screenshot as: task2_verify_group.png



sudo cat /etc/shadow

- Save screenshot as: task2_verify_shadow.png



Task 3 – Create groups; change tom's primary and secondary groups

Goal: Create groups developer, devops, and designer. Change tom's primary group and manage secondary groups.

1. Create groups and verify by viewing /etc/group (visually confirm entries exist):

sudo groupadd developer

sudo groupadd devops

sudo groupadd designer

cat /etc/group

- Save screenshot as: task3_groupadd.png

```
hamail@ubuntu-lab:~$ sudo groupadd developer
sudo groupadd devops
sudo groupadd designer
cat /etc/group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog,hamail
tty:x:5:
disk:x:6:
lp:x:7:
mail:x:8:
news:x:9:
uucp:x:10:
man:x:12:
proxy:x:13:
kmem:x:15:
dialout:x:20:
fax:x:21:
voice:x:22:
cdrom:x:24:hamail
floppy:x:25:
tape:x:26:
sudo:x:27:hamail
audio:x:29:pulse
dip:x:30:hamail
www-data:x:33:
```

2. Change tom's primary group to designer and verify:

sudo usermod -g designer tom

id tom

- Save screenshot as: task3_change_primary_group.png

```
hamail@ubuntu-lab:~$ sudo usermod -g designer tom
id tom
uid=1001(tom) gid=1004(designer) groups=1004(designer),100(users)
hamail@ubuntu-lab:~$
```

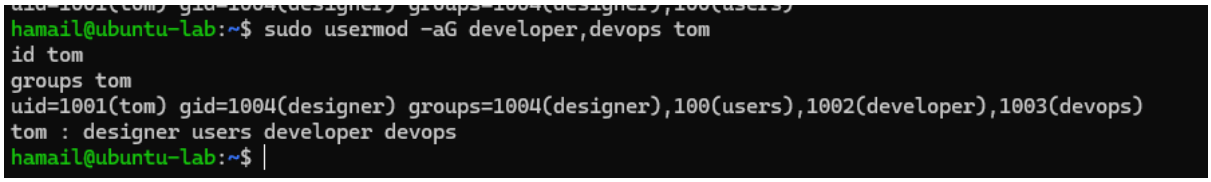
3. Add secondary groups developer and devops to tom and verify:

sudo usermod -aG developer,devops tom

id tom

groups tom

- Save screenshot as: task3_add_secondary_groups.png



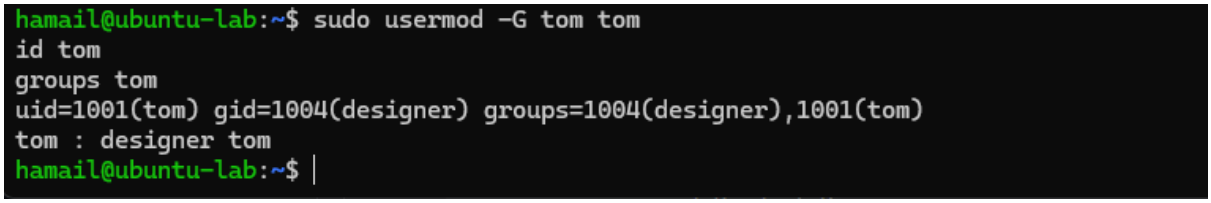
4. Replace all secondary groups so only tom (user's own group) remains and verify:

sudo usermod -G tom tom

id tom

groups tom

- Save screenshot as: task3_reset_secondary_groups.png



Task 4 – Create/delete users (Jerry, Scooby) and groups (jolly, anime)

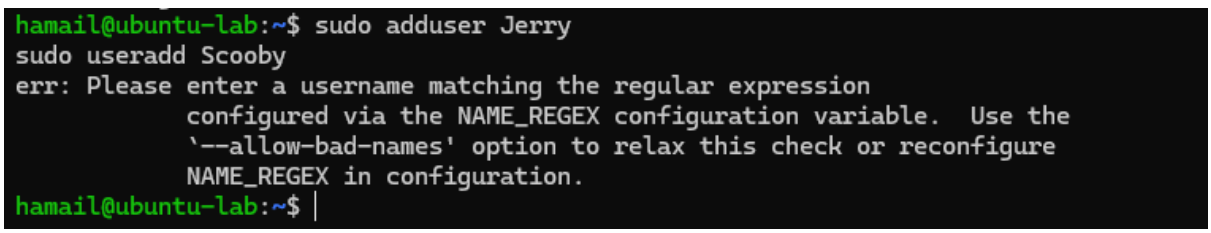
Goal: Create users using both adduser and useradd, demonstrate login/password/home directory differences, then delete users/groups.

1. Create users:

sudo adduser Jerry

sudo useradd Scooby

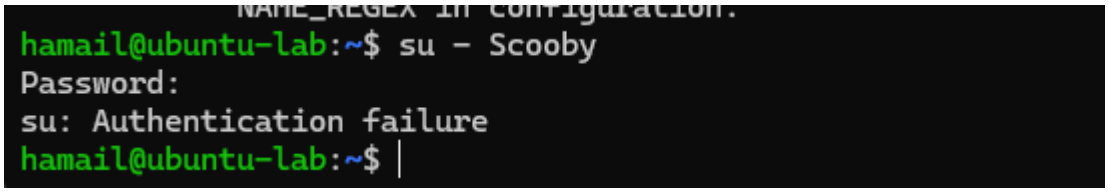
- Save screenshot as: task4_add_users.png



2. Try to log in as Scooby immediately (expected authentication failure because there is no password yet):

su - Scooby

- Save screenshot as: task4_scooby_su_auth_failure.png



3. Set a password for Scooby:

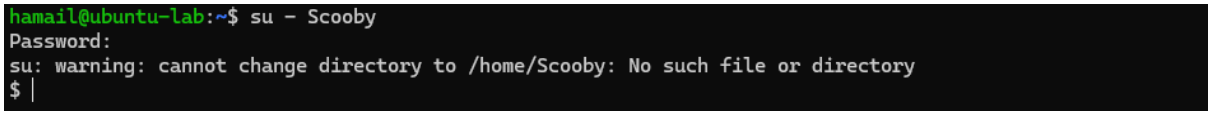
sudo passwd Scooby

- Save screenshot as: task4_set_password_scooby.png

4. Try logging in as Scooby again (home directory still missing; expect a message such as "No directory, logging in with HOME=/"):

su - Scooby

- Save screenshot as: task4_scooby_su_no_home.png



5. Show that Scooby's home directory does not exist yet and what /etc/passwd says:

exit

cat /etc/passwd

ls -ld /home/Scooby

- Save screenshot as: task4_scooby_no_home.png

6. Manually create Scooby's home directory and set proper ownership and permissions:

sudo mkdir -p /home/Scooby

sudo chown Scooby:Scooby /home/Scooby

sudo chmod 750 /home/Scooby

ls -ld /home/Scooby

- Save screenshot as: task4_scooby_create_home.png

```
hamail@ubuntu-lab:~$ sudo mkdir -p /home/Scooby
sudo chown Scooby:Scooby /home/Scooby
sudo chmod 750 /home/Scooby
ls -ld /home/Scooby
drwxr-x--- 2 Scooby Scooby 4096 Nov  2 10:34 /home/Scooby
hamail@ubuntu-lab:~$
```

7. Log in as Scooby again and verify you land in the correct home directory:

su - Scooby

pwd

ls -la

- Save screenshot as: task4_scooby_login_success.png

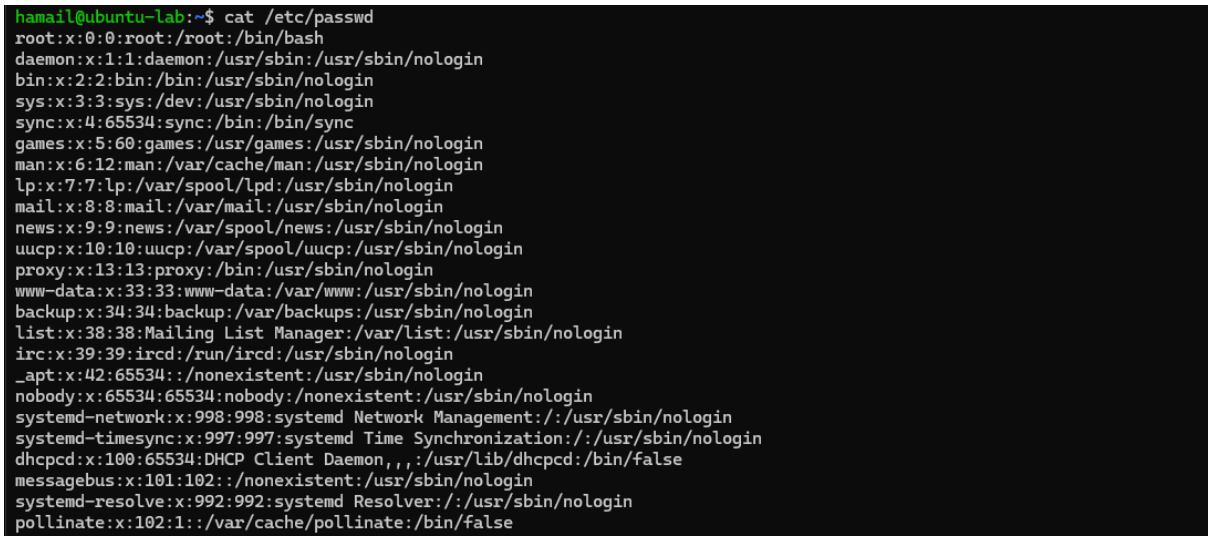
```
hamail@ubuntu-lab:~$ su - Scooby
pwd
ls -la
Password:
$
$
$
$
$ su - Scooby
pwd
Password:
su: Authentication failure
$
$
$
$ exit
cat /etc/pa/home/hamail
total 411792
drwxr-x--- 23 hamail hamail      4096 Nov  1 19:15 .
drwxr-xr-x  5 root   root        4096 Nov  2 10:34 ..
-rw-rw-r--  1 hamail hamail       557 Oct 31 19:14 answers.md
-rw-rw-r--  1 hamail hamail       269 Nov  1 13:19 apt_update_vs_upgrade.md
-rw-------  1 hamail hamail      6089 Nov  1 19:18 .bash_history
-rw-r--r--  1 hamail hamail       220 Mar 31  2024 .bash_logout
-rw-r--r--  1 hamail hamail      3771 Mar 31  2024 .bashrc
drwx------  7 hamail hamail      4096 Nov  1 14:42 .cache
drwx------  8 hamail hamail      4096 Nov  1 14:42 .config
drwxr-xr-x  2 hamail hamail      4096 Nov  1 14:17 Desktop
```

8. Verify users in system files and observe shell of Scooby:

exit

cat /etc/passwd

- Save screenshot as: task4_verify_users.png



9. Change the shell from /bin/sh to /bin/bash

sudo usermod -s /bin/bash Scooby

su - Scooby

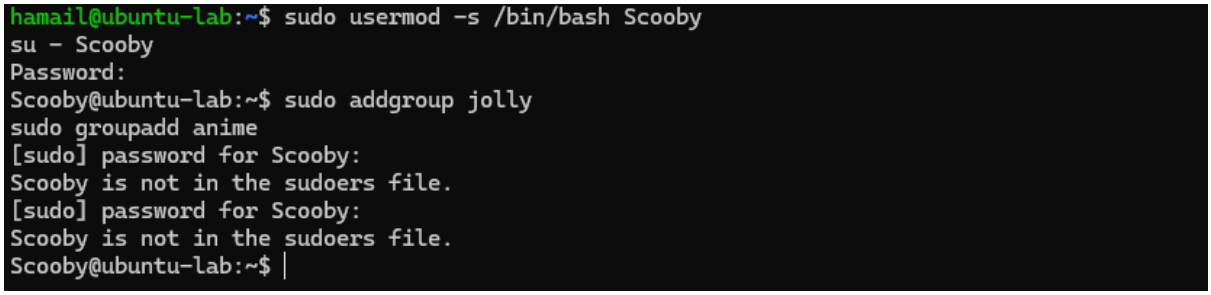
- Save screenshot as: task4_shell_switching.png

10. Create groups:

sudo addgroup jolly

sudo groupadd anime

- Save screenshot as: task4_add_groups.png



10. Verify groups:

cat /etc/group

- Save screenshot as: task4_verify_groups.png

```
hamail@ubuntu-lab:~$ sudo usermod -s /bin/bash Scooby
su - Scooby
Password:
Scooby@ubuntu-lab:~$ sudo addgroup jolly
sudo groupadd anime
[sudo] password for Scooby:
Scooby is not in the sudoers file.
[sudo] password for Scooby:
Scooby is not in the sudoers file.
Scooby@ubuntu-lab:~$ cat /etc/group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog,hamail
tty:x:5:
disk:x:6:
lp:x:7:
mail:x:8:
news:x:9:
uucp:x:10:
man:x:12:
proxy:x:13:
kmem:x:15:
dialout:x:20:
fax:x:21:
voice:x:22:
```

11. Delete groups and users:

sudo delgroup jolly

sudo groupdel anime

cat /etc/group

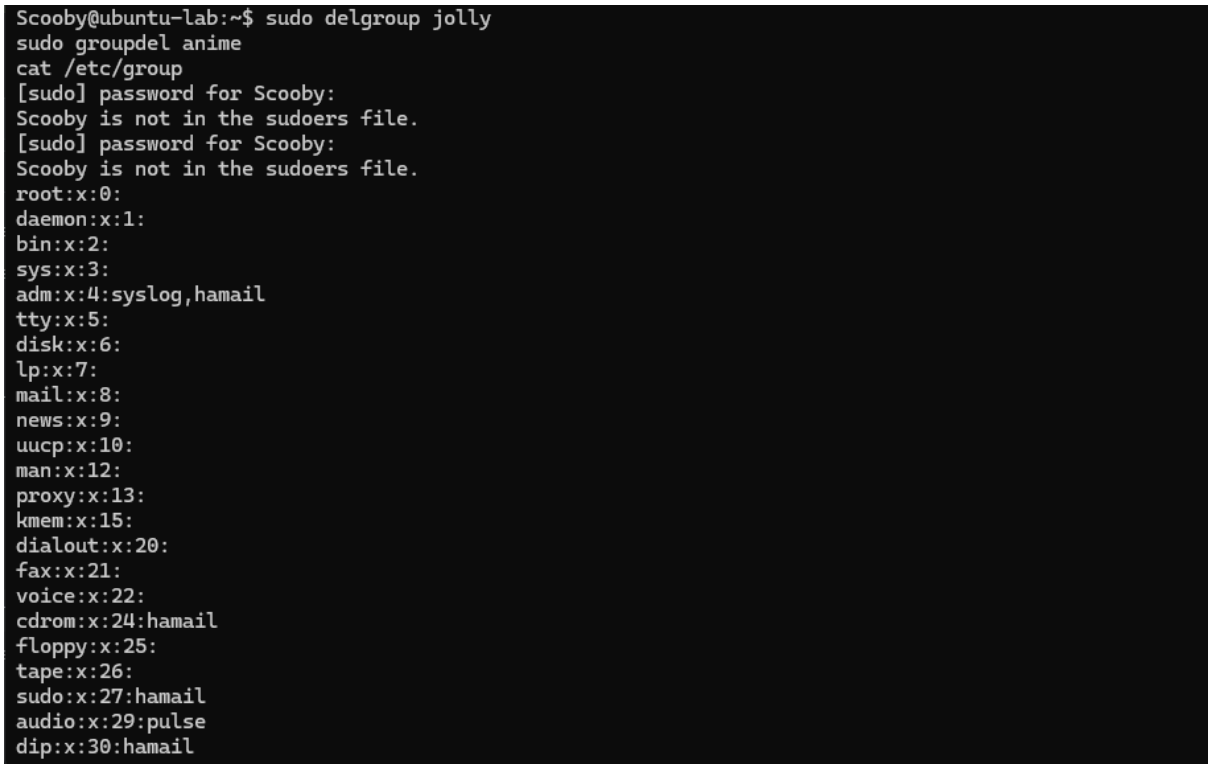

```
hamail@ubuntu-lab:~$ sudo usermod -s /bin/bash Scooby
su - Scooby
Password:
Scooby@ubuntu-lab:~$ sudo addgroup jolly
sudo groupadd anime
[sudo] password for Scooby:
Scooby is not in the sudoers file.
[sudo] password for Scooby:
Scooby is not in the sudoers file.
Scooby@ubuntu-lab:~$ cat /etc/group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog,hamail
tty:x:5:
disk:x:6:
lp:x:7:
mail:x:8:
news:x:9:
uucp:x:10:
man:x:12:
proxy:x:13:
kmem:x:15:
dialout:x:20:
```

sudo deluser --remove-home Jerry

sudo userdel -r Scooby

cat /etc/passwd

- Save screenshots as: task4_delete_groups.png, task4_delete_users.png

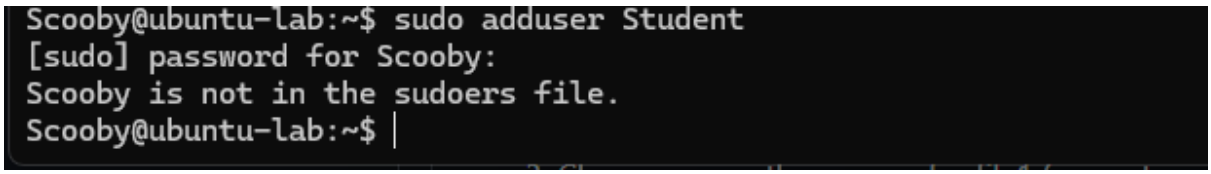


Task 5 – Create user Student; create files; set owner/group; identify file types

1. Create Student:

   sudo adduser Student

- Save screenshot as: task5_create_student.png



2. Switch to Student and create files:

   su - Student

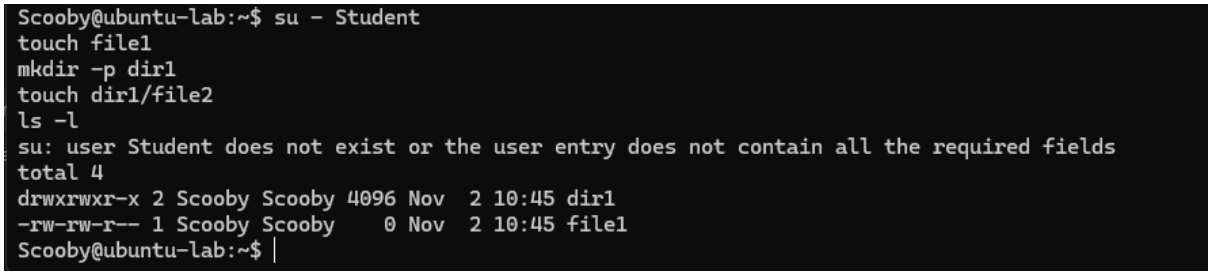
   touch file1

   mkdir -p dir1

   touch dir1/file2

ls -l

- Save screenshot as: task5_create_files.png



3. Change owner then group for file1 (separate commands):

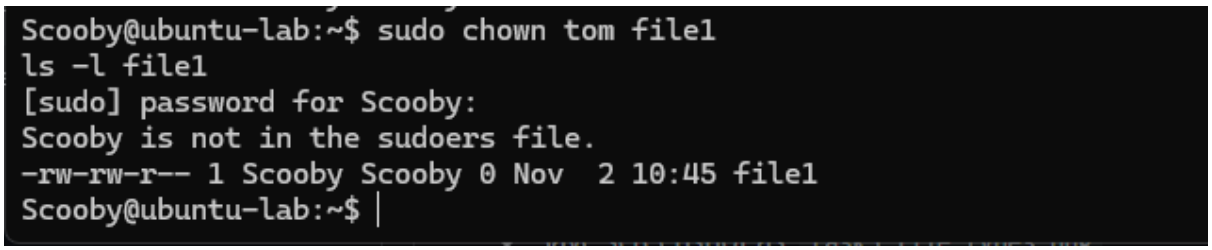
   sudo chown tom file1

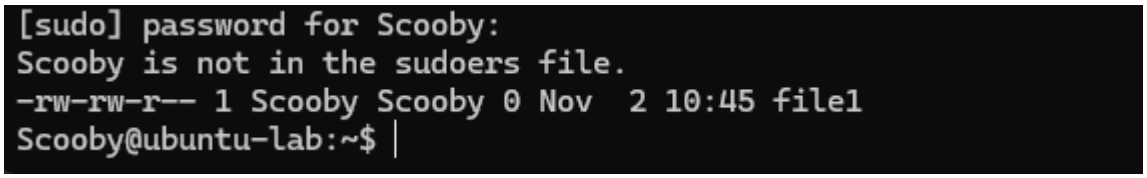
   ls -l file1

   sudo chgrp devops file1

   ls -l file1

- Save screenshots as: task5_chown_file1.png, task5_chgrp_file1.png





4. Identify files/directories and show /dev/null:

   ls -l

   ls -l dir1

   ls -l /dev/null

   file file1 dir1 /dev/null

- Save screenshot as: task5_file_types.png

```
Scooby@ubuntu-lab:~$ ls -l
ls -l dir1
ls -l /dev/null
file file1 dir1 /dev/null
total 4
drwxrwxr-x 2 Scooby Scooby 4096 Nov  2 10:45 dir1
-rw-rw-r-- 1 Scooby Scooby    0 Nov  2 10:45 file1
total 0
-rw-rw-r-- 1 Scooby Scooby 0 Nov  2 10:45 file2
crw-rw-rw- 1 root root 1, 3 Nov  2 10:02 /dev/null
file1:     empty
dir1:      directory
/dev/null: character special (1/3)
Scooby@ubuntu-lab:~$
```

5. Exit Student:

   exit

- Save screenshot as: task5_exit_student.png

```
Scooby@ubuntu-lab:~$ exit
logout
hamail@ubuntu-lab:~$
```

Task 6 – Change permissions using symbolic mode

Target file: ~/file1 (run these as the Student user)

1. Ensure Student and file present:

   su - Student

   cd ~

   ls -l file1

- Save screenshot as: task6_su_student.png

```
hamail@ubuntu-lab:~$ su - Student
cd ~
ls -l file1
su: user Student does not exist or the user entry does not contain all the required fields
ls: cannot access 'file1': No such file or directory
hamail@ubuntu-lab:~$
```

2. Remove all permissions:

   chmod -rwx file1

   ls -l file1

- Save screenshot as: task6_chmod_remove_rwx.png

```
hamail@ubuntu-lab:~$ chmod -rwx file1
ls -l file1
---------- 1 hamail hamail 7 Nov  2 10:54 file1
hamail@ubuntu-lab:~$
```

3. Add read to all:

   chmod +r file1

   ls -l file1

- Save screenshot as: task6_chmod_add_r.png

```
hamail@ubuntu-lab:~$ chmod +r file1
ls -l file1
-r--r--r-- 1 hamail hamail 7 Nov  2 10:54 file1
hamail@ubuntu-lab:~$
```

4. Add execute to user:

   chmod u+x file1

   ls -l file1

- Save screenshot as: task6_chmod_u_plus_x.png

```
hamail@ubuntu-lab:~$ chmod u+x file1
ls -l file1
-r-xr--r-- 1 hamail hamail 7 Nov  2 10:54 file1
hamail@ubuntu-lab:~$
```

5. Add write to user and group:

   chmod ug+w file1

   ls -l file1

- Save screenshot as: task6_chmod_ug_plus_w.png

```
hamail@ubuntu-lab:~$ chmod ug+w file1
ls -l file1
-rwxrw-r-- 1 hamail hamail 7 Nov  2 10:54 file1
hamail@ubuntu-lab:~$
```

6. Remove all permissions (explicit):

   chmod ugo-rwx file1

   ls -l file1

- Save screenshot as: task6_chmod_ugo_minus_rwx.png

```
hamail@ubuntu-lab:~$ chmod ugo-rwx file1
ls -l file1
---------- 1 hamail hamail 7 Nov  2 10:54 file1
hamail@ubuntu-lab:~$
```

Task 7 – Change permissions using "set" symbolic form (u= g= o=)

Ensure you are Student:

su - Student

cd ~

ls -l file1

- Save screenshot as: task7_student_context.png

```
hamail@ubuntu-lab:~$ su - Student   # or skip if already Student
cd ~
ls -l file1
su: user Student does not exist or the user entry does not contain all the required fields
---------- 1 hamail hamail 7 Nov  2 10:54 file1
hamail@ubuntu-lab:~$
```

1. Set all to rwx:

   chmod u=rwx,g=rwx,o=rwx file1

   ls -l file1

- Save screenshot as: task7_chmod_set_all_rwx.png

```
hamail@ubuntu-lab:~$ chmod u=rwx,g=rwx,o=rwx file1
ls -l file1
-rwxrwxrwx 1 hamail hamail 7 Nov  2 10:54 file1
hamail@ubuntu-lab:~$
```

2. Remove execute from group and others:

   chmod g=rw,o=rw file1

   ls -l file1

- Save screenshot as: task7_remove_exec_go.png

```
hamail@ubuntu-lab:~$ chmod g=rw,o=rw file1
ls -l file1
-rwxrw-rw- 1 hamail hamail 7 Nov  2 10:54 file1
hamail@ubuntu-lab:~$
```

3. Remove all permissions:

chmod u=,g=,o= file1

ls -l file1

- Save screenshot as: task7_remove_all_perms.png

```
hamail@ubuntu-lab:~$ chmod u=,g=,o= file1
ls -l file1
---------- 1 hamail hamail 7 Nov  2 10:54 file1
hamail@ubuntu-lab:~$
```

Task 8 – Change permissions using numeric (octal) mode

Ensure you are Student:

su - Student

cd ~

ls -l file1

- Save screenshot as: task8_student_context.png

Run each command and capture screenshot after each ls:

```
hamail@ubuntu-lab:~$ su - Student
cd ~
ls -l file1
su: user Student does not exist or the user entry does not contain all the required fields
---------- 1 hamail hamail 7 Nov  2 10:54 file1
hamail@ubuntu-lab:~$
```

1.

chmod 777 file1

ls -l file1

- task8_chmod_777.png

```
hamail@ubuntu-lab:~$ chmod 777 file1
ls -l file1
-rwxrwxrwx 1 hamail hamail 7 Nov  2 10:54 file1
hamail@ubuntu-lab:~$
```

2.

chmod 700 file1

ls -l file1

- task8_chmod_700.png

```
hamail@ubuntu-lab:~$ chmod 700 file1
ls -l file1
-rwx------ 1 hamail hamail 7 Nov  2 10:54 file1
hamail@ubuntu-lab:~$
```

3.

chmod 744 file1

ls -l file1

- task8_chmod_744.png

```
hamail@ubuntu-lab:~$ chmod 744 file1
ls -l file1
-rwxr--r-- 1 hamail hamail 7 Nov  2 10:54 file1
hamail@ubuntu-lab:~$
```

4.

chmod 640 file1

ls -l file1

- task8_chmod_640.png

```
hamail@ubuntu-lab:~$ chmod 640 file1
ls -l file1
-rw-r----- 1 hamail hamail 7 Nov  2 10:54 file1
hamail@ubuntu-lab:~$
```

5.

chmod 664 file1

ls -l file1

- task8_chmod_664.png

```
hamail@ubuntu-lab:~$ chmod 664 file1
ls -l file1
-rw-rw-r-- 1 hamail hamail 7 Nov  2 10:54 file1
hamail@ubuntu-lab:~$
```

6.

chmod 775 file1

ls -l file1

- task8_chmod_775.png

```
hamail@ubuntu-lab:~$ chmod 775 file1
ls -l file1
-rwxrwxr-x 1 hamail hamail 7 Nov  2 10:54 file1
hamail@ubuntu-lab:~$
```

7.

chmod 750 file1

ls -l file1

- task8_chmod_750.png

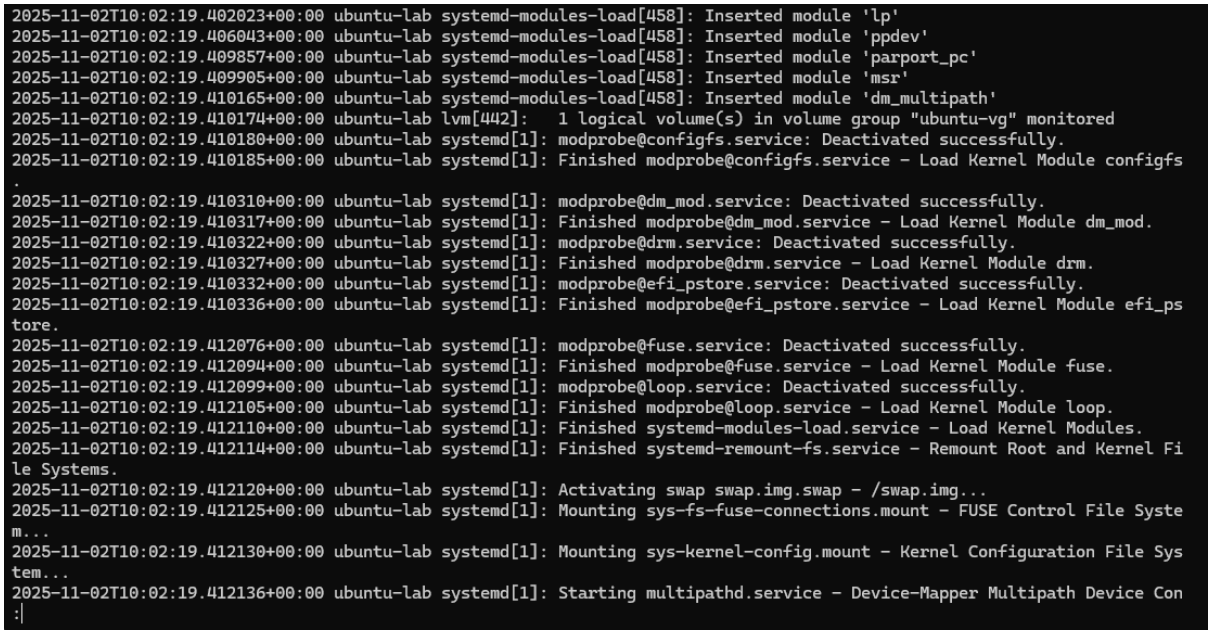
Task 9 – Practice pipes, pagers, grep, and redirects with /var/log/syslog

1. less:

   sudo cat /var/log/syslog | less

   # quit q

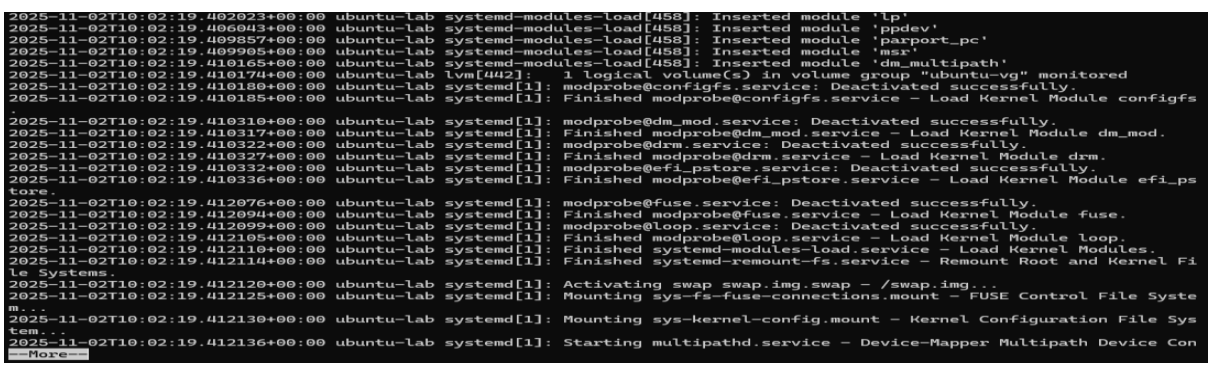
- task9_grep_less.png



2. more:

   sudo cat /var/log/syslog | more

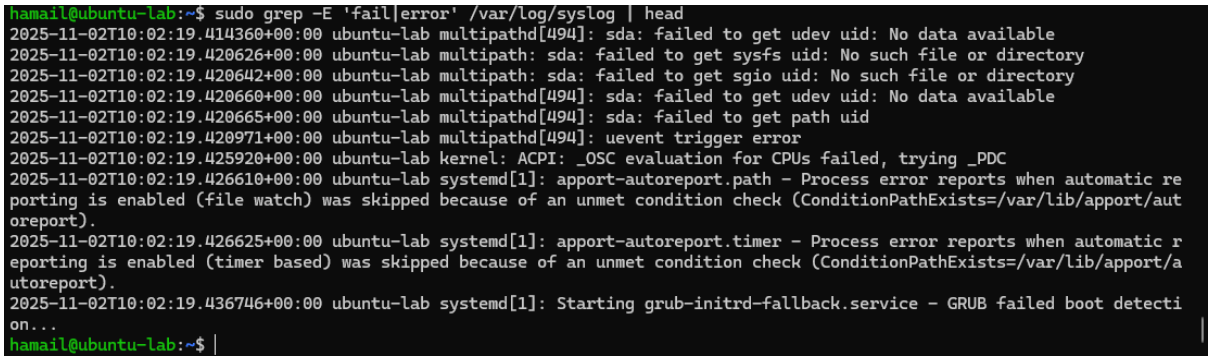
- task9_grep_more.png



3. grep failures/errors:

   sudo grep -E 'fail|error' /var/log/syslog | head

- task9_grep_head.png



4. redirect:

   sudo grep -i systemd /var/log/syslog > ~/syslog_systemd.txt

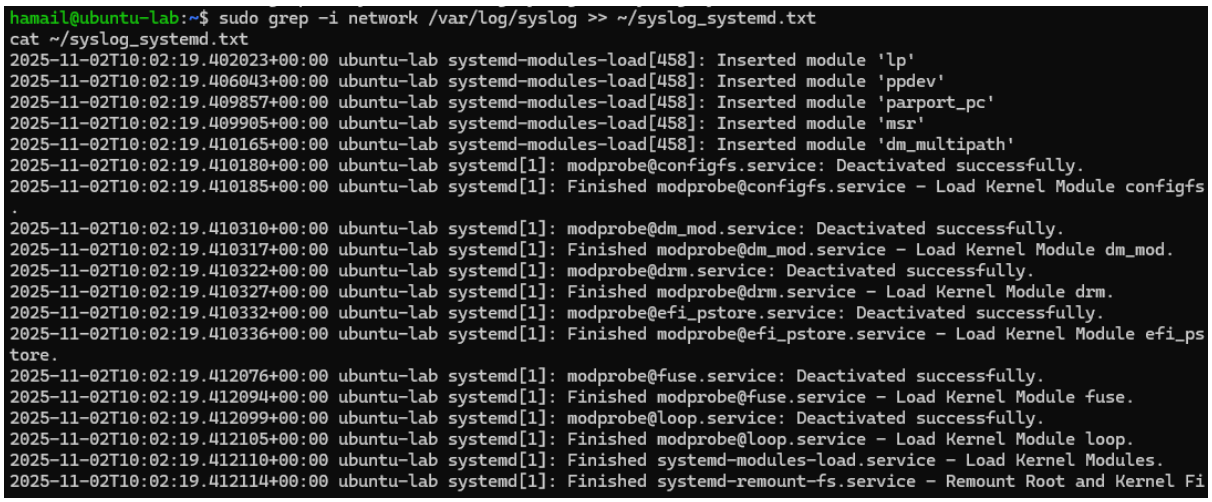
- task9_redirect_overwrite.png

   append:

   sudo grep -i network /var/log/syslog >> ~/syslog_systemd.txt

   cat ~/syslog_systemd.txt

- task9_redirect_append.png

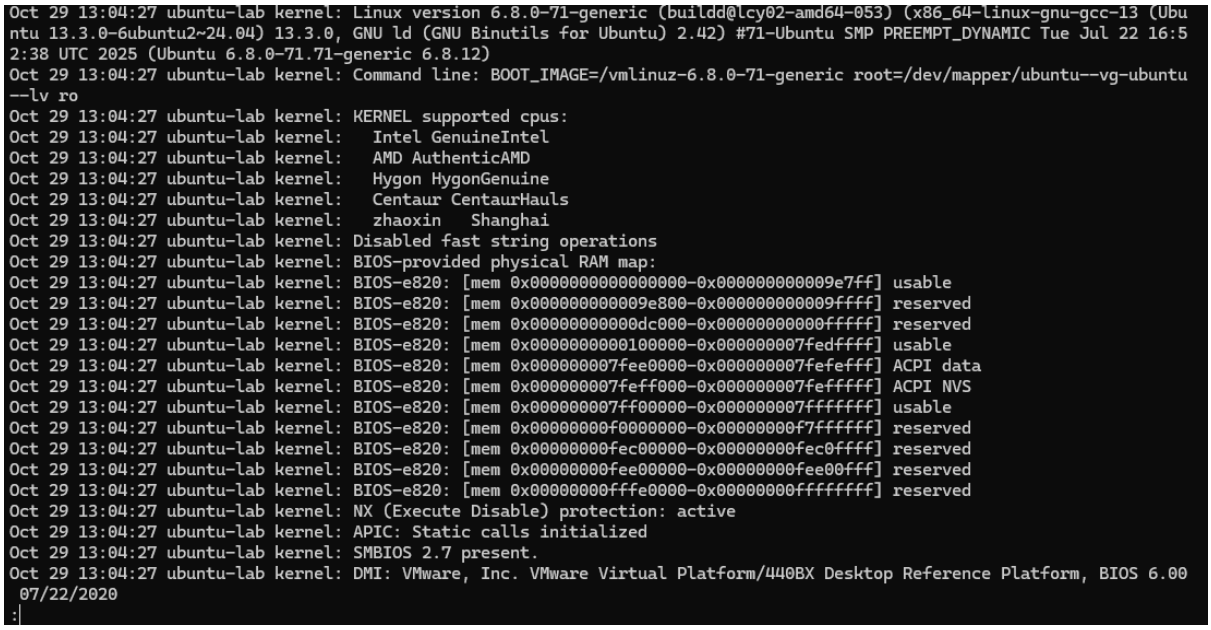


   Alternative (journalctl) if needed:

   sudo journalctl | less

   sudo journalctl -u systemd | grep -i error > ~/journal_errors.txt

- task9_journalctl_alternative.png



Task 10 – Script setup.sh – variables, command substitution, file/dir checks, permissions (use vim)

Goal: Using vim, write a script named setup.sh that implements each numbered step below. After writing the code for each step, run the script and capture screenshots showing the vim editor (script content) and the script output for that step. Students must add the code for each step into the same file setup.sh step-by-step (i.e., write 1., save, run and screenshot; then append 2., save, run and screenshot; and so on).

For each step you MUST:

- Open vim and edit setup.sh
- Insert only the code shown for that step (append to the existing file)
- Save and quit vim (:wq)
- Make the file executable if not already: chmod +x setup.sh
- Run the script: ./setup.sh
- Capture two screenshots:
  - One showing the vim editor with the script content after you added the step (use the vim screen before :wq)
  - One showing the terminal output after running the script (show the command and the output)

Start in your Student home directory (recommended).

1. Include bash shebang

- Code to add (enter in vim as the first line of the file):

  #!/bin/bash

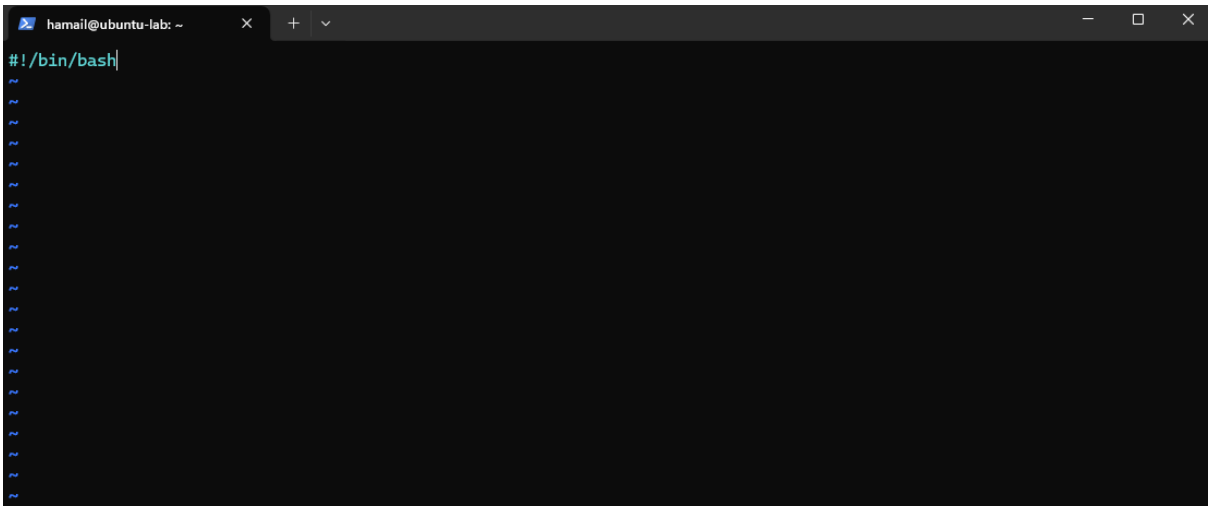
- Steps:

i. vim setup.sh → add the shebang line → save and quit

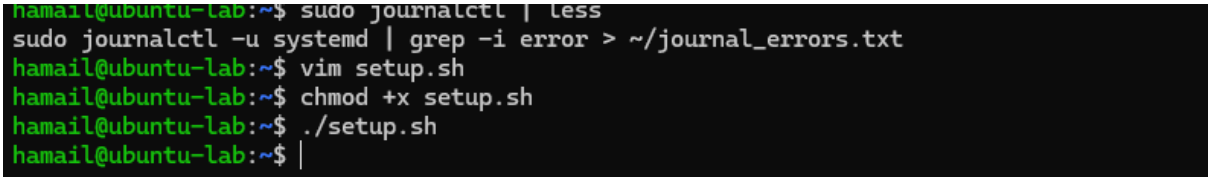
ii. chmod +x setup.sh

iii. ./setup.sh

- Screenshots:
  - vim editor showing the shebang: task10_b1_vim.png



  - script run output (likely no output but show ./setup.sh run): task10_b1_run.png



2. Define variable var1 and echo it

- Code to append:

  # Define and show var1

  var1="Hello from Lab 6"

  echo "var1: $var1"

- Steps:

i. vim setup.sh → append the code above → save and quit

ii. ./setup.sh

- Screenshots:

  - vim editor showing var1 code appended: task10_b2_vim.png

  - script run output showing var1 printed: task10_b2_run.png

3. Save output of ls -l into variable allFiles and echo it

- Code to append:

  # Save ls -l to variable and display

  allFiles="$(ls -l)"

  echo "allFiles (ls -l):"

  echo "$allFiles"

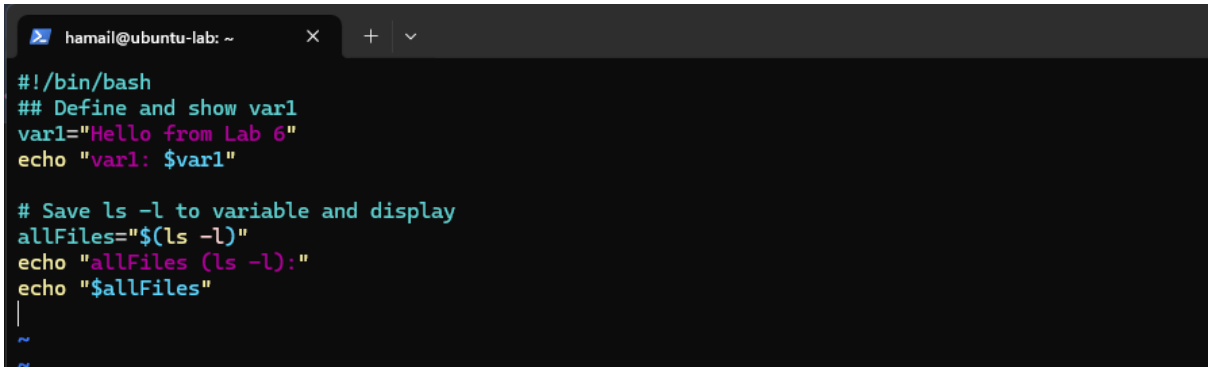
- Steps:

i. vim setup.sh → append the code above → save and quit

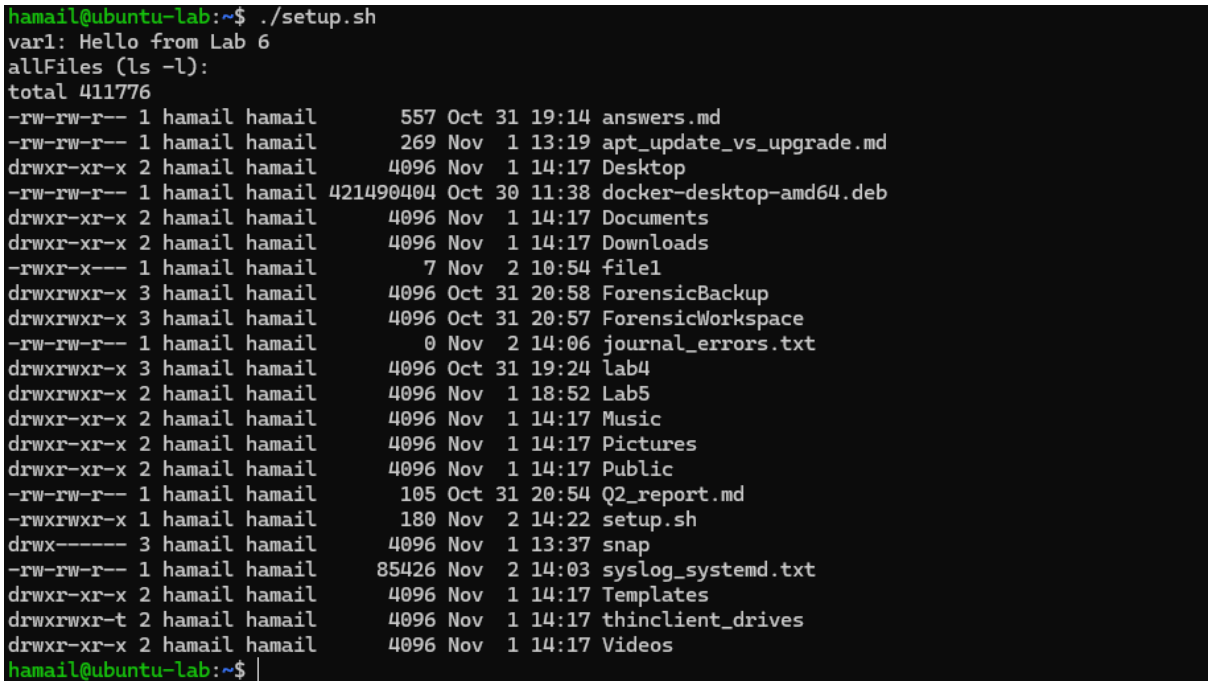
ii. ./setup.sh

- Screenshots:

  - vim editor showing allFiles code appended: task10_b3_vim.png



- script run output showing the ls -l content echoed: task10_b3_run.png



4. If directory dir1 exists echo a message; else create it

- Code to append:

```
# Directory check
if [ -d "dir1" ]; then
  echo "Directory dir1 exists."
else
  echo "Directory dir1 does not exist. Creating..."
  mkdir -p "dir1"
  echo "Directory dir1 created."
fi
```

- Steps:

i. vim setup.sh → append the code above → save and quit

ii. ./setup.sh

- Screenshots:

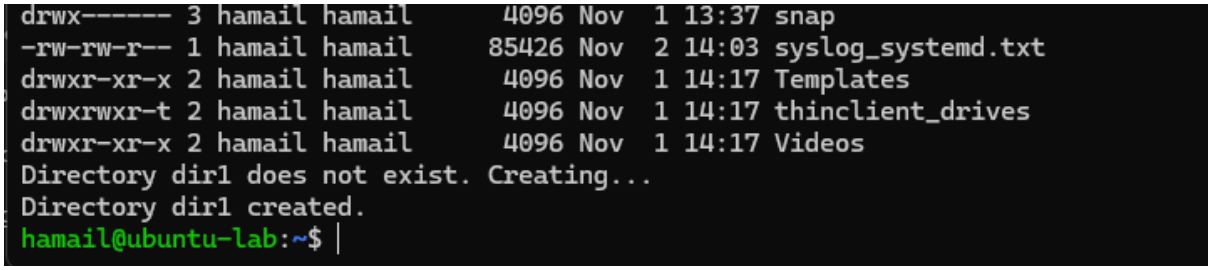
- vim editor showing dir1 check code: task10_b4_vim.png

```
#!/bin/bash
## Define and show var1
var1="Hello from Lab 6"
echo "var1: $var1"

# Save ls -l to variable and display
allFiles="$(ls -l)"
echo "allFiles (ls -l):"
echo "$allFiles"

# Directory check
if [ -d "dir1" ]; then
  echo "Directory dir1 exists."
else
  echo "Directory dir1 does not exist. Creating..."
  mkdir -p "dir1"
  echo "Directory dir1 created."
fi
```

- script run output showing directory message or creation: task10_b4_run.png



5. If file dir1/file2 does not exist, create it

- Code to append:

```
# File check
if [ -f "dir1/file2" ]; then
  echo "file2 already exists."
else
  echo "file2 does not exist. Creating..."
  touch "dir1/file2"
  chmod a-rwx "dir1/file2"
  echo "file2 created."
fi
```

- Steps:

i. vim setup.sh → append the code above → save and quit

ii. ./setup.sh

- Screenshots:

- vim editor showing file2 check code: task10_b5_vim.png

```
#!/bin/bash
## Define and show var1
var1="Hello from Lab 6"
echo "var1: $var1"

# Save ls -l to variable and display
allFiles="$(ls -l)"
echo "allFiles (ls -l):"
echo "$allFiles"

# Directory check
if [ -d "dir1" ]; then
  echo "Directory dir1 exists."
else
  echo "Directory dir1 does not exist. Creating..."
  mkdir -p "dir1"
  echo "Directory dir1 created."
fi

# File check
if [ -f "dir1/file2" ]; then
  echo "file2 already exists."
else
  echo "file2 does not exist. Creating..."
  touch "dir1/file2"
  chmod a-rwx "dir1/file2"
  echo "file2 created."
fi
```

- script run output showing file creation message or existence: task10_b5_run.png

```
var1: Hello from Lab 6
allFiles (ls -l):
total 411780
-rw-rw-r-- 1 hamail hamail       557 Oct 31 19:14 answers.md
-rw-rw-r-- 1 hamail hamail       269 Nov  1 13:19 apt_update_vs_upgrade.md
drwxr-xr-x 2 hamail hamail      4096 Nov  1 14:17 Desktop
drwxrwxr-x 2 hamail hamail      4096 Nov  2 14:26 dir1
-rw-rw-r-- 1 hamail hamail 421490404 Oct 30 11:38 docker-desktop-amd64.deb
drwxr-xr-x 2 hamail hamail      4096 Nov  1 14:17 Documents
drwxr-xr-x 2 hamail hamail      4096 Nov  1 14:17 Downloads
-rwxr-x--- 1 hamail hamail         7 Nov  2 10:54 file1
drwxrwxr-x 3 hamail hamail      4096 Oct 31 20:58 ForensicBackup
drwxrwxr-x 3 hamail hamail      4096 Oct 31 20:57 ForensicWorkspace
-rw-rw-r-- 1 hamail hamail         0 Nov  2 14:06 journal_errors.txt
drwxrwxr-x 3 hamail hamail      4096 Oct 31 19:24 lab4
drwxrwxr-x 2 hamail hamail      4096 Nov  1 18:52 Lab5
drwxr-xr-x 2 hamail hamail      4096 Nov  1 14:17 Music
drwxr-xr-x 2 hamail hamail      4096 Nov  1 14:17 Pictures
drwxr-xr-x 2 hamail hamail      4096 Nov  1 14:17 Public
-rw-rw-r-- 1 hamail hamail       105 Oct 31 20:54 Q2_report.md
-rwxrwxr-x 1 hamail hamail       562 Nov  2 14:27 setup.sh
drwx------ 3 hamail hamail      4096 Nov  1 13:37 snap
-rw-rw-r-- 1 hamail hamail     85426 Nov  2 14:03 syslog_systemd.txt
drwxr-xr-x 2 hamail hamail      4096 Nov  1 14:17 Templates
drwxrwxr-t 2 hamail hamail      4096 Nov  1 14:17 thinclient_drives
drwxr-xr-x 2 hamail hamail      4096 Nov  1 14:17 Videos
Directory dir1 exists.
file2 does not exist. Creating...
file2 created.
```

6. Check read, write, execute permissions on dir1/file2; grant missing user perms and show final ls

- Code to append:

  # Permission checks for dir1/file2 (user permissions)

```
f="dir1/file2"
if [ ! -r "$f" ]; then
  echo "Read permission missing; granting to user..."
  chmod u+r "$f"
fi

if [ ! -w "$f" ]; then
  echo "Write permission missing; granting to user..."
  chmod u+w "$f"
fi

if [ ! -x "$f" ]; then
  echo "Execute permission missing; granting to user..."
  chmod u+x "$f"
fi

echo "Final permissions for $f:"
ls -l "$f"
```

- Steps:

i. vim setup.sh → append the code above → save and quit

ii. ./setup.sh

- Screenshots:
  - vim editor showing permission-check code: task10_b6_vim.png

```
# Permission checks for dir1/file2 (user permissions)
f="dir1/file2"
if [ ! -r "$f" ]; then
  echo "Read permission missing; granting to user..."
  chmod u+r "$f"
fi

if [ ! -w "$f" ]; then
  echo "Write permission missing; granting to user..."
  chmod u+w "$f"
fi

if [ ! -x "$f" ]; then
  echo "Execute permission missing; granting to user..."
  chmod u+x "$f"
fi

echo "Final permissions for $f:"
ls -l "$f"
```

- script run output showing the permission grants and final ls -l dir1/file2: task10_b6_run.png

```
Directory dir1 exists.
file2 already exists.
Read permission missing; granting to user...
Write permission missing; granting to user...
Execute permission missing; granting to user...
Final permissions for dir1/file2:
-rwx------ 1 hamail hamail 0 Nov  2 14:28 dir1/file2
hamail@ubuntu-lab:~$
```

Task 11 – Script setup.sh – argument comparisons (eq, ne, gt, lt, ge, le) and string checks

Updated: replace the previous single-script approach with an incremental exercise. Students will overwrite setup.sh and then add each individual if-test one-by-one. After adding each if-test they must run the script with example arguments and capture screenshots. This teaches the individual comparison operators and makes each if statement a separate step.

Important overall instructions

- Start by overwriting setup.sh (vim setup.sh) and add only what the step asks (do not add all tests at once).
- After editing in vim, save (:wq), make executable (chmod +x setup.sh) if needed, then run the script with the example commands shown for each step.
- For each step capture two screenshots:
- A vim screenshot showing the current file buffer with the newly added lines (before :wq) — name as specified for the step.

- A terminal screenshot showing the commands you ran (chmod +x setup.sh if necessary) and the script outputs for the example invocations — name as specified for the step.

- For the numeric comparisons, set a variable num=$1 at the top of the file before adding the individual if-tests (this will be the initial step). For string checks, set str=$2 before adding the string if-tests.

1. create file with shebang and set num and str variables

- In vim create/overwrite setup.sh and insert:

  #!/bin/bash

  num=$1

  str=$2

- Save and quit (:wq)

- Make executable and run with examples:

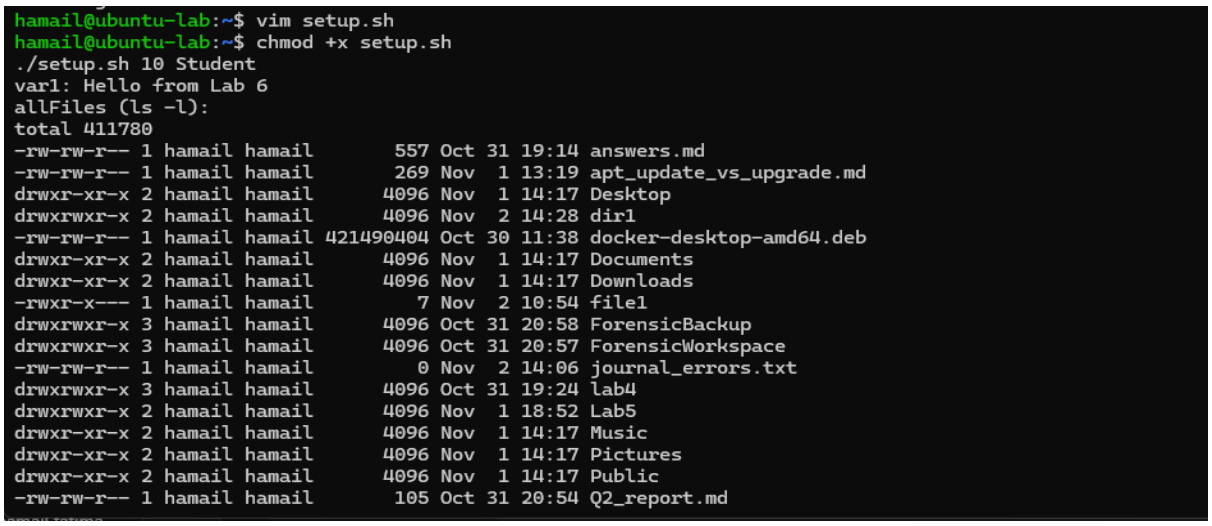
  chmod +x setup.sh

  ./setup.sh 10 Student

- Screenshots:

  - vim content: task11_b0_vim.png

  - run output: task11_b0_run.png



2. add the -eq test (equal)

- Append to setup.sh:

  if ["$num" -eq 10]; then

  echo "$num is equal to 10 (-eq)."

  else

  echo "$num is NOT equal to 10 (-eq)."

  fi

- Save and quit; then run these commands (capture both in one terminal screenshot):

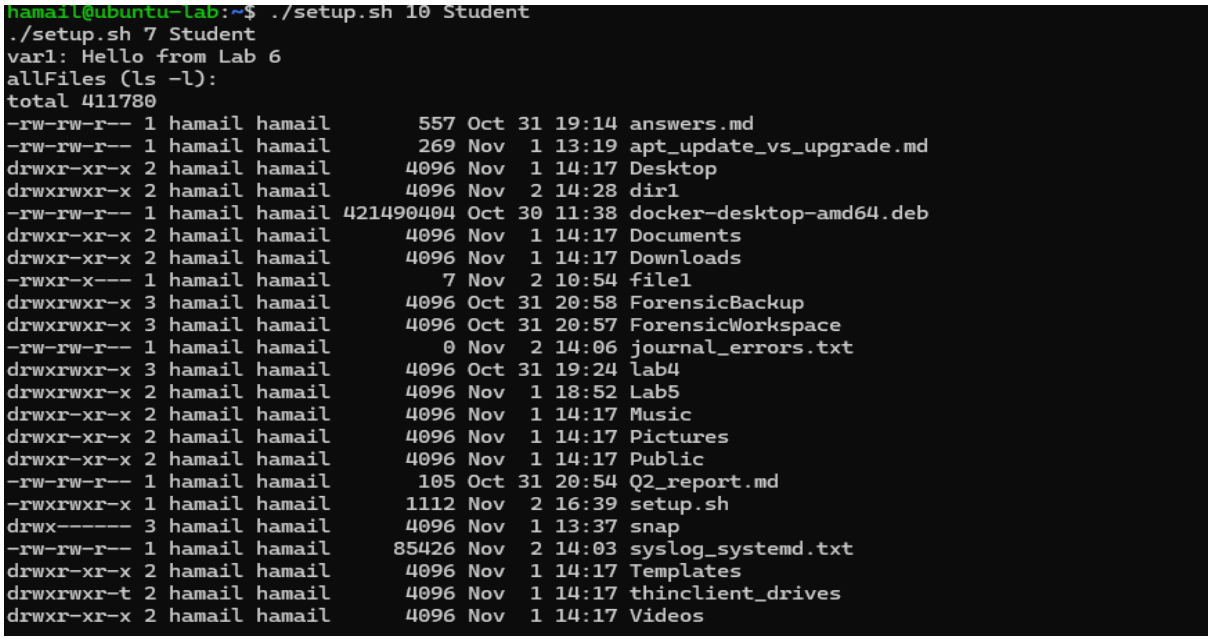
  ./setup.sh 10 Student

  ./setup.sh 7 Student

- Screenshots:

  - vim content after edit: task11_b1_vim.png

  - run output demonstrating both cases: task11_b1_run.png



3. add the -ne test (not equal)
    - Append to setup.sh:

    if ["$num" -ne 10]; then

      echo "$num is not equal to 10 (-ne)."

    else

      echo "$num is equal to 10 (-ne false)."

    fi

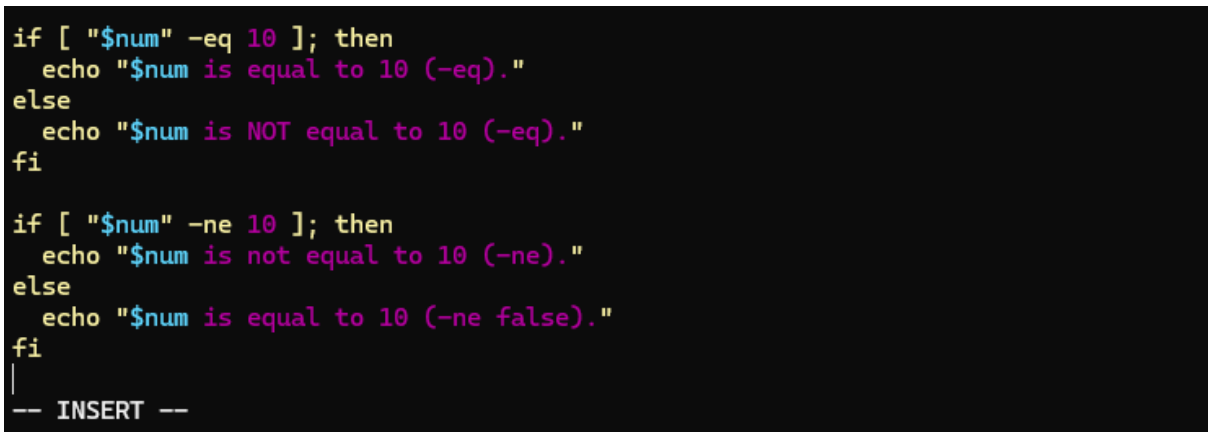
    - Save and quit; run:

    ./setup.sh 7 Student

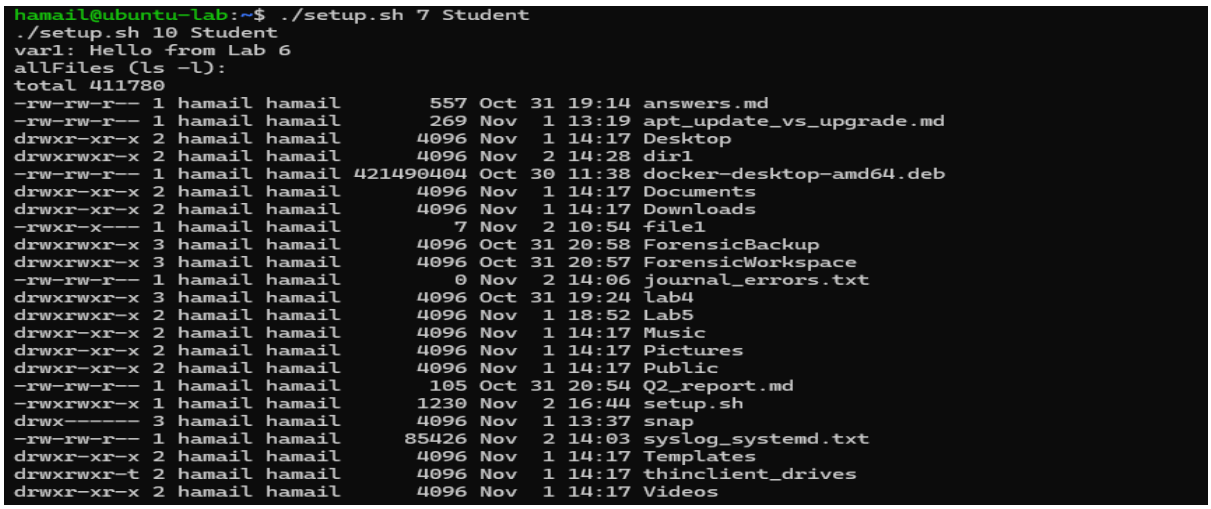
    ./setup.sh 10 Student

    - Screenshots:

    - vim content: task11_b2_vim.png



    - run output: task11_b2_run.png



4. add the -gt test (greater than)

- Append:

if ["$num" -gt 10]; then

echo "$num is greater than 10 (-gt)."

else

echo "$num is NOT greater than 10 (-gt)."

fi

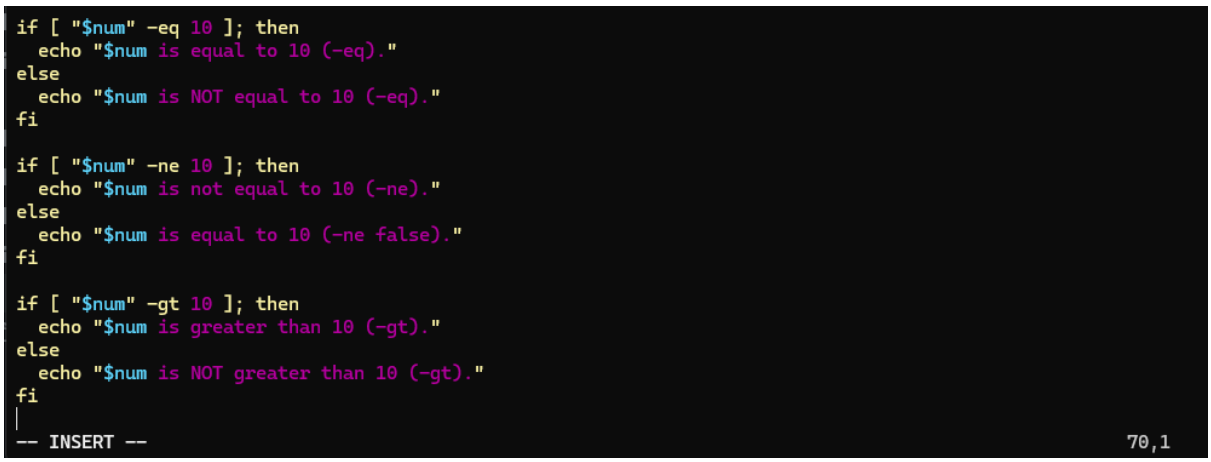
- Run:

./setup.sh 12 Student

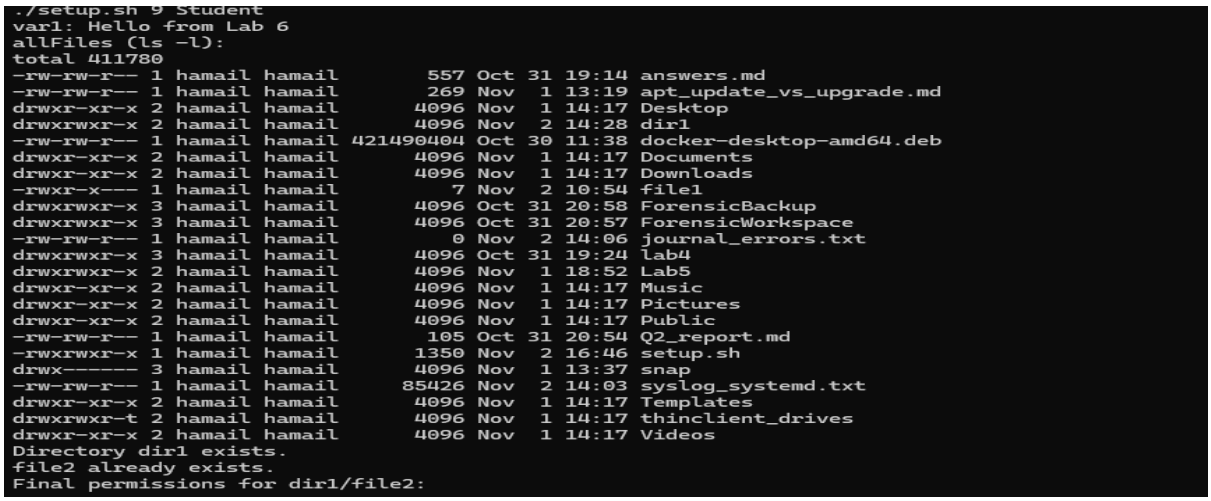
./setup.sh 9 Student

- Screenshots:

- vim content: task11_b3_vim.png



- run output: task11_b3_run.png



5. add the -lt test (less than)

- Append:

if ["$num" -lt 10]; then

echo "$num is less than 10 (-lt)."

else

echo "$num is NOT less than 10 (-lt)."

fi

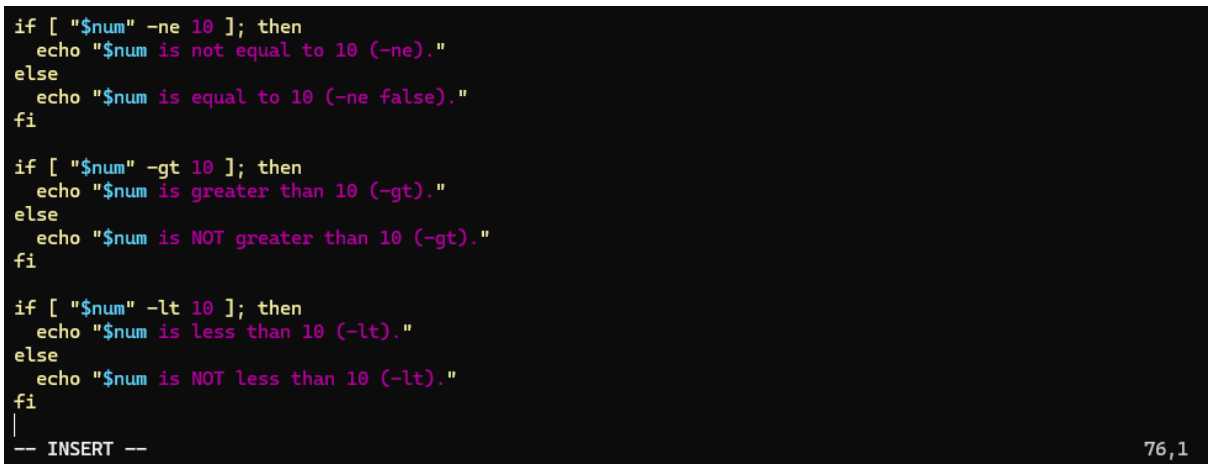
- Run:

./setup.sh 5 Student

./setup.sh 11 Student

- Screenshots:

- vim content: task11_b4_vim.png



- run output: task11_b4_run.png

6. add the -ge test (greater than or equal)

- Append:

if ["$num" -ge 10]; then

echo "$num is greater than or equal to 10 (-ge)."

else

echo "$num is NOT greater than or equal to 10 (-ge)."

fi

- Run:

./setup.sh 10 Student

./setup.sh 8 Student

- Screenshots:

- vim content: task11_b5_vim.png

```
if [ "$num" -gt 10 ]; then
  echo "$num is greater than 10 (-gt)."
else
  echo "$num is NOT greater than 10 (-gt)."
fi

if [ "$num" -lt 10 ]; then
  echo "$num is less than 10 (-lt)."
else
  echo "$num is NOT less than 10 (-lt)."
fi

if [ "$num" -ge 10 ]; then
  echo "$num is greater than or equal to 10 (-ge)."
else
  echo "$num is NOT greater than or equal to 10 (-ge)."
fi
-- INSERT --
```

- run output: task11_b5_run.png

```
Directory dir1 exists.
file2 already exists.
Final permissions for dir1/file2:
-rwx------ 1 hamail hamail 0 Nov  2 14:28 dir1/file2
8 is NOT equal to 10 (-eq).
8 is not equal to 10 (-ne).
8 is NOT greater than 10 (-gt).
8 is less than 10 (-lt).
8 is NOT greater than or equal to 10 (-ge).
hamail@ubuntu-lab:~$
```

7. add the -le test (less than or equal)

- Append:

if ["$num" -le 10]; then

echo "$num is less than or equal to 10 (-le)."

else

echo "$num is NOT less than or equal to 10 (-le)."

fi

- Run:

./setup.sh 10 Student

./setup.sh 12 Student

- Screenshots:

- vim content: task11_b6_vim.png

```
if [ "$num" -ge 10 ]; then
  echo "$num is greater than or equal to 10 (-ge)."
else
  echo "$num is NOT greater than or equal to 10 (-ge)."
fi

if [ "$num" -le 10 ]; then
  echo "$num is less than or equal to 10 (-le)."
else
  echo "$num is NOT less than or equal to 10 (-le)."
fi

-- INSERT --
```

- run output: task11_b6_run.png

```
Directory dir1 exists.
file2 already exists.
Final permissions for dir1/file2:
-rwx------ 1 hamail hamail 0 Nov  2 14:28 dir1/file2
12 is NOT equal to 10 (-eq).
12 is not equal to 10 (-ne).
12 is greater than 10 (-gt).
12 is NOT less than 10 (-lt).
12 is greater than or equal to 10 (-ge).
12 is NOT less than or equal to 10 (-le).
hamail@ubuntu-lab:~$
```

8. string equality test (=)

- Ensure str=$2 exists at top (1.). Append:

if ["$str" = "Student"]; then

echo "Second argument equals 'Student' (=)."

else

echo "Second argument does NOT equal 'Student' (=)."

fi

- Run:

./setup.sh 10 Student

./setup.sh 10 Test

- Screenshots:

- vim content: task11_b7_vim.png

```
if [ "$num" -le 10 ]; then
  echo "$num is less than or equal to 10 (-le)."
else
  echo "$num is NOT less than or equal to 10 (-le)."
fi

if [ "$str" = "Student" ]; then
  echo "Second argument equals 'Student' ( = )."
else
  echo "Second argument does NOT equal 'Student' ( = )."
fi
-- INSERT --
```

- run output: task11_b7_run.png

```
Directory dir1 exists.
file2 already exists.
Final permissions for dir1/file2:
-rwx------ 1 hamail hamail 0 Nov  2 14:28 dir1/file2
10 is equal to 10 (-eq).
10 is equal to 10 (-ne false).
10 is NOT greater than 10 (-gt).
10 is NOT less than 10 (-lt).
10 is greater than or equal to 10 (-ge).
10 is less than or equal to 10 (-le).
Second argument does NOT equal 'Student' ( = ).
hamail@ubuntu-lab:~$
```

9. string inequality test (!=)

- Append:

if ["$str" != "Student"]; then

  echo "Second argument is not equal to 'Student' (!=)."

else

  echo "Second argument equals 'Student' (!= false)."

fi

- Run:

./setup.sh 10 Test

./setup.sh 10 Student

- Screenshots:

- vim content: task11_b8_vim.png

```
if [ "$num" -le 10 ]; then
  echo "$num is less than or equal to 10 (-le)."
else
  echo "$num is NOT less than or equal to 10 (-le)."
fi

if [ "$str" = "Student" ]; then
  echo "Second argument equals 'Student' ( = )."
else
  echo "Second argument does NOT equal 'Student' ( = )."
fi

if [ "$str" != "Student" ]; then
  echo "Second argument is not equal to 'Student' ( != )."
else
  echo "Second argument equals 'Student' ( != false)."
fi
```

- run output: task11_b8_run.png

```
Directory dir1 exists.
file2 already exists.
Final permissions for dir1/file2:
-rwx------ 1 hamail hamail 0 Nov  2 14:28 dir1/file2
10 is equal to 10 (-eq).
10 is equal to 10 (-ne false).
10 is NOT greater than 10 (-gt).
10 is NOT less than 10 (-lt).
10 is greater than or equal to 10 (-ge).
10 is less than or equal to 10 (-le).
Second argument equals 'Student' ( = ).
Second argument equals 'Student' ( != false).
hamail@ubuntu-lab:~$
```

10. check if second argument is empty (zero-length)

- Append:

```
if [ -z "$str" ]; then
  echo "Second argument is empty (zero-length)."
else
  echo "Second argument is not empty."
fi
```

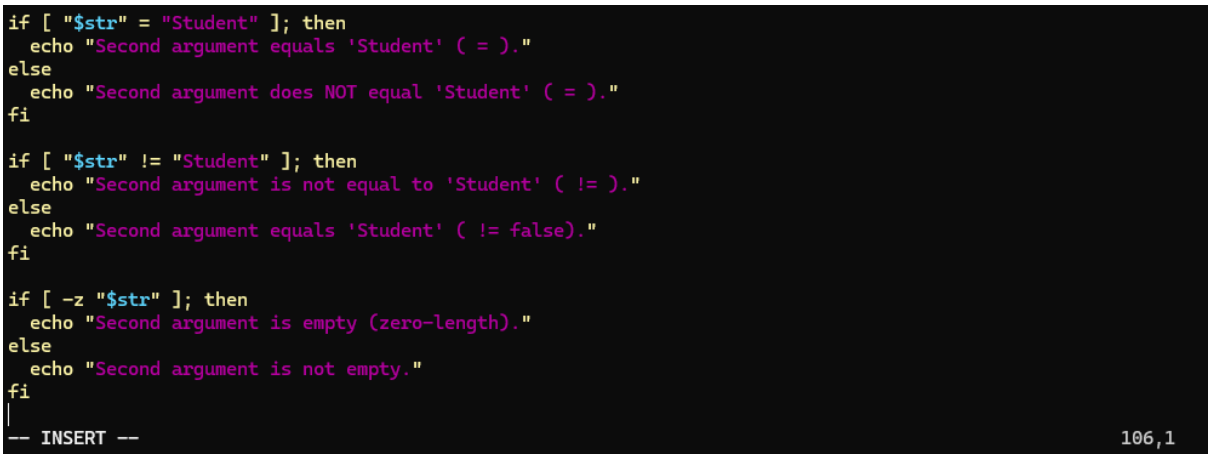
- Run:

./setup.sh 10

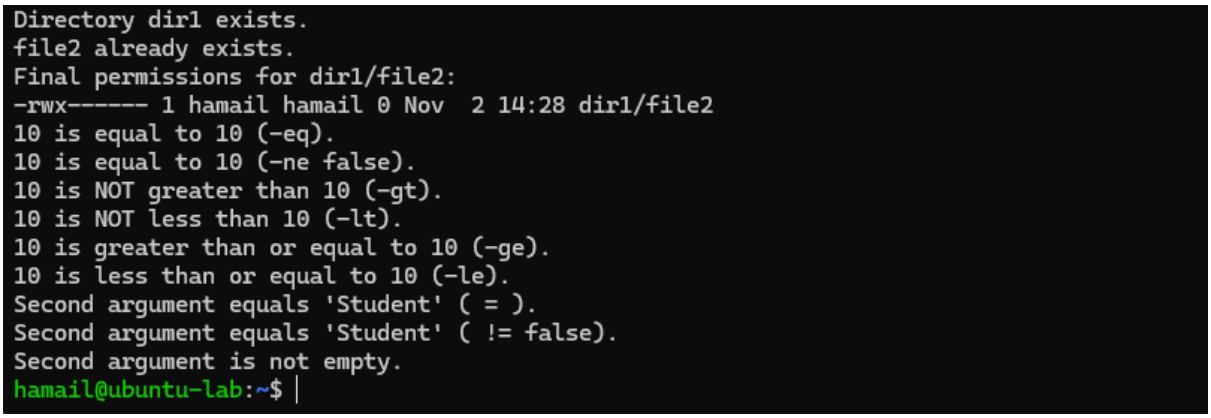
./setup.sh 10 Student

- Screenshots:

- vim content: task11_b9_vim.png



- run output: task11_b9_run.png



Task 12 – Script setup.sh – print all arguments with a for loop

1. Create the script with shebang and basic structure

- Open vim and overwrite setup.sh:

  vim setup.sh

- Insert these lines (first step — shebang and a short comment):

  #!/bin/bash

  # Script to demonstrate printing all user-entered arguments using $*

- Save and quit (:wq)
- Screenshots:

- vim editor showing the shebang and comment: task12_b1_vim.png

```
fi
#!/bin/bash
# Script to demonstrate printing all user-entered arguments using $*

-- INSERT --
```

- run (no output expected but show ./setup.sh run): task12_b1_run.png

```
file2 already exists.
Final permissions for dir1/file2:
-rwx------ 1 hamail hamail 0 Nov  2 14:28 dir1/file2
./setup.sh: line 53: [: : integer expression expected
 is NOT equal to 10 (-eq).
./setup.sh: line 59: [: : integer expression expected
 is equal to 10 (-ne false).
./setup.sh: line 65: [: : integer expression expected
 is NOT greater than 10 (-gt).
./setup.sh: line 71: [: : integer expression expected
 is NOT less than 10 (-lt).
./setup.sh: line 77: [: : integer expression expected
 is NOT greater than or equal to 10 (-ge).
./setup.sh: line 83: [: : integer expression expected
 is NOT less than or equal to 10 (-le).
Second argument does NOT equal 'Student' ( = ).
Second argument is not equal to 'Student' ( != ).
Second argument is empty (zero-length).
hamail@ubuntu-lab:~$
```

2. Append the for loop using $* and print each argument

- Re-open setup.sh in vim and append the following lines:

  # Print all arguments using $*

  echo "Printing all arguments using \$*:"

  for arg in $*; do

    echo "Argument: $arg"

  done

- Save and quit (:wq)
- Make the script executable and run it with example arguments:

  chmod +x setup.sh

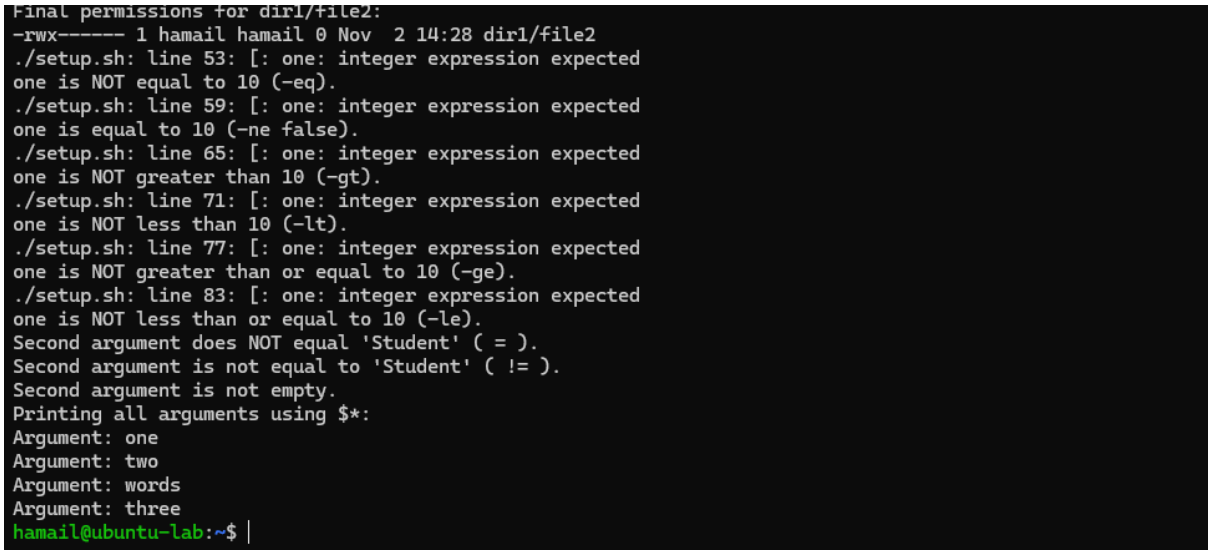
  ./setup.sh one "two words" three

- Screenshots:

- vim editor showing the for-loop appended: task12_b2_vim.png

```
#!/bin/bash
# Script to demonstrate printing all user-entered arguments using $*
# Print all arguments using $*
echo "Printing all arguments using \$*:"
for arg in $*; do
  echo "Argument: $arg"
done
```

- script run output showing the printed arguments: task12_b2_run.png



Task 13 – Script setup.sh – while loop summation and functions

Clear the previous code of setup.sh and write a new script, step-by-step, that:

- Starts with a shebang line
- Implements an interactive while loop that prompts the user to enter numbers and keeps a running total until the user types q to quit; after each input the script echoes "Total Score: <current_total>"
- Implements a function sum_two() that runs its own interactive while loop doing the same accumulation and echoes the running totals
- Adds a second function that takes two numeric arguments, sums them, and returns the result via echo (demonstrated in the script)
- Important: if you move the while-loop logic into the sum_two() function, delete the standalone while-loop code to avoid running the same loop twice

1. Add the shebang line

- Open vim and overwrite setup.sh with the shebang line:

  #!/bin/bash

- Save and quit (:wq)

- Make executable and run (no output expected):

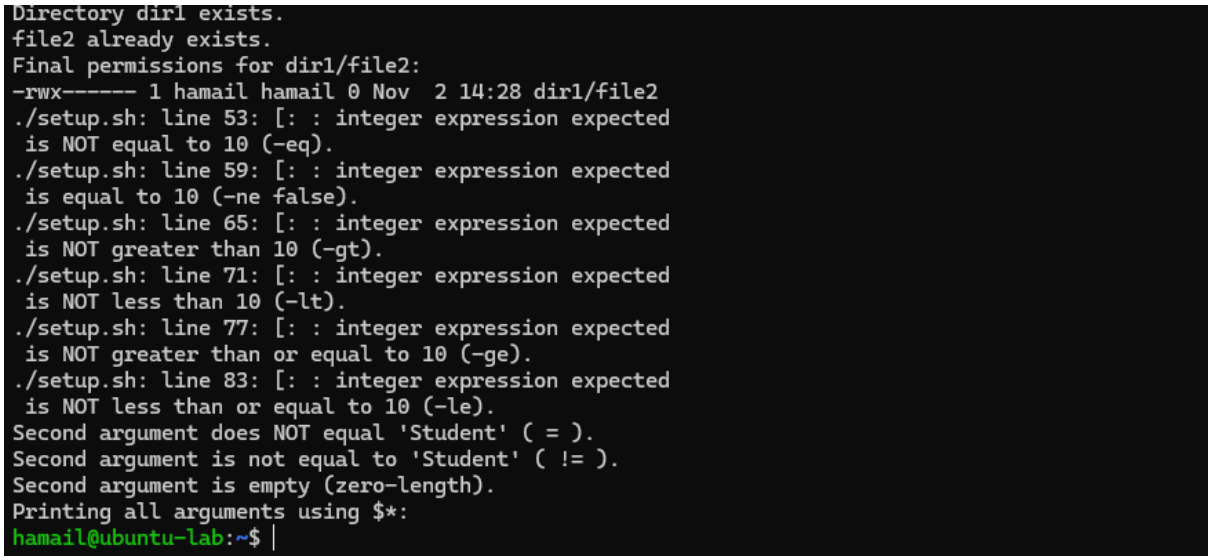
  chmod +x setup.sh

  ./setup.sh

- Screenshots:

  - vim editor showing shebang: task13_b1_vim.png

  - run output: task13_b1_run.png



2. Add the while-loop summation (interactive)

- Re-open setup.sh in vim and append the while-loop:

  # While-loop summation (interactive)

  sum=0

  while true; do

  read -p "Enter a number (or 'q' to quit): " input

  if ["$input" = "q"]; then

  break

  fi

  sum=$((sum + input))

  echo "Total Score: $sum"

done

echo "Final total: $sum"

- Save and quit (:wq)
- Run the script and demonstrate a short session (example): enter 5, then 7, then q

  ./setup.sh

  # interactively enter:

  # 5

  # 7

  # q
- Screenshots:
  - vim editor showing while-loop appended: task13_b2_vim.png

```
# Script to demonstrate printing all user-entered arguments using $*
# Print all arguments using $*
echo "Printing all arguments using \$*:"
for arg in $*; do
  echo "Argument: $arg"
done
#!/bin/bash
## While-loop summation (interactive)
sum=0
while true; do
  read -p "Enter a number (or 'q' to quit): " input
  if [ "$input" = "q" ]; then
    break
  fi

  # validate numeric input to avoid errors:
  if ! [[ "$input" =~ ^-?[0-9]+$ ]]; then
    echo "Please enter an integer or 'q'."
    continue
  fi

  sum=$((sum + input))
  echo "Total Score: $sum"
done
echo "Final total: $sum"
-- INSERT --
```

  - run output showing the interactive session and totals: task13_b2_run.png

```
Directory dir1 exists.
file2 already exists.
Final permissions for dir1/file2:
-rwx------ 1 hamail hamail 0 Nov  2 14:28 dir1/file2
./setup.sh: line 53: [: : integer expression expected
 is NOT equal to 10 (-eq).
./setup.sh: line 59: [: : integer expression expected
 is equal to 10 (-ne false).
./setup.sh: line 65: [: : integer expression expected
 is NOT greater than 10 (-gt).
./setup.sh: line 71: [: : integer expression expected
 is NOT less than 10 (-lt).
./setup.sh: line 77: [: : integer expression expected
 is NOT greater than or equal to 10 (-ge).
./setup.sh: line 83: [: : integer expression expected
 is NOT less than or equal to 10 (-le).
Second argument does NOT equal 'Student' ( = ).
Second argument is not equal to 'Student' ( != ).
Second argument is empty (zero-length).
Printing all arguments using $*:
Enter a number (or 'q' to quit):
```

3. Add the interactive summation function and demonstrate it

- Re-open setup.sh in vim and append the function sum_two() which contains its own interactive while-loop:

```
# Function to accumulate scores interactively
sum_two() {
 sum=0
 while true; do
  read -p "Enter a number (or 'q' to quit): " input
  if [ "$input" = "q" ]; then
   break
  fi
  sum=$((sum + input))
  echo "Total Score: $sum"
 done
 echo "Function final total: $sum"
}
# Demonstrate the function
echo "Now calling sum_two function:"
sum_two
```

- Save and quit (:wq)
- Important: If you have the standalone while-loop from step 2 and you place this function into the script, delete the standalone loop to avoid executing the same interactive logic twice when running the script.
- Run the script and demonstrate a short session (example): enter 3, 4, q when prompted by the function:

```
./setup.sh
# when prompted by the function enter:
# 3
# 4
```

q

- Screenshots:

- vim editor showing function appended: task13_b3_vim.png

```
  sum=$((sum + input))
  echo "Total Score: $sum"
done
echo "Final total: $sum"
# Function to accumulate scores interactively
sum_two() {
  sum=0
  while true; do
    read -p "Enter a number (or 'q' to quit): " input
    if [ "$input" = "q" ]; then
      break
    fi

    if ! [[ "$input" =~ ^-?[0-9]+$ ]]; then
      echo "Please enter an integer or 'q'."
      continue
    fi

    sum=$((sum + input))
    echo "Total Score: $sum"
  done
  echo "Function final total: $sum"
}

# Demonstrate the function
echo "Now calling sum_two function:"
sum_two
```

- run output showing the function prompts and final total: task13_b3_run.png

```
Directory dir1 exists.
file2 already exists.
Final permissions for dir1/file2:
-rwx------ 1 hamail hamail 0 Nov  2 14:28 dir1/file2
./setup.sh: line 53: [: : integer expression expected
 is NOT equal to 10 (-eq).
./setup.sh: line 59: [: : integer expression expected
 is equal to 10 (-ne false).
./setup.sh: line 65: [: : integer expression expected
 is NOT greater than 10 (-gt).
./setup.sh: line 71: [: : integer expression expected
 is NOT less than 10 (-lt).
./setup.sh: line 77: [: : integer expression expected
 is NOT greater than or equal to 10 (-ge).
./setup.sh: line 83: [: : integer expression expected
 is NOT less than or equal to 10 (-le).
Second argument does NOT equal 'Student' ( = ).
Second argument is not equal to 'Student' ( != ).
Second argument is empty (zero-length).
Printing all arguments using $*:
Enter a number (or 'q' to quit):
```

4. Add a function that takes two numeric arguments, sums them, and returns the result (echo)

- Re-open setup.sh in vim and append the following function and demonstration. This function accepts two numeric arguments, adds them, and return the sum. The script then captures that output and displays it.

  # Function that sums two arguments and returns the result

  sum_args() {

   a=$1

   b=$2

   return $((a + b))

  }

  # Demonstrate sum_args function

  echo "Now demonstrating sum_args function:"

  sum_args 3 4

  result=$?

  echo "sum_args(3,4) returned: $result"

  - Save and quit (:wq)
  - Run the script and capture the demonstration output:

  chmod +x setup.sh

  ./setup.sh

  # Observe the output that shows "sum_args(3,4) returned: 7"

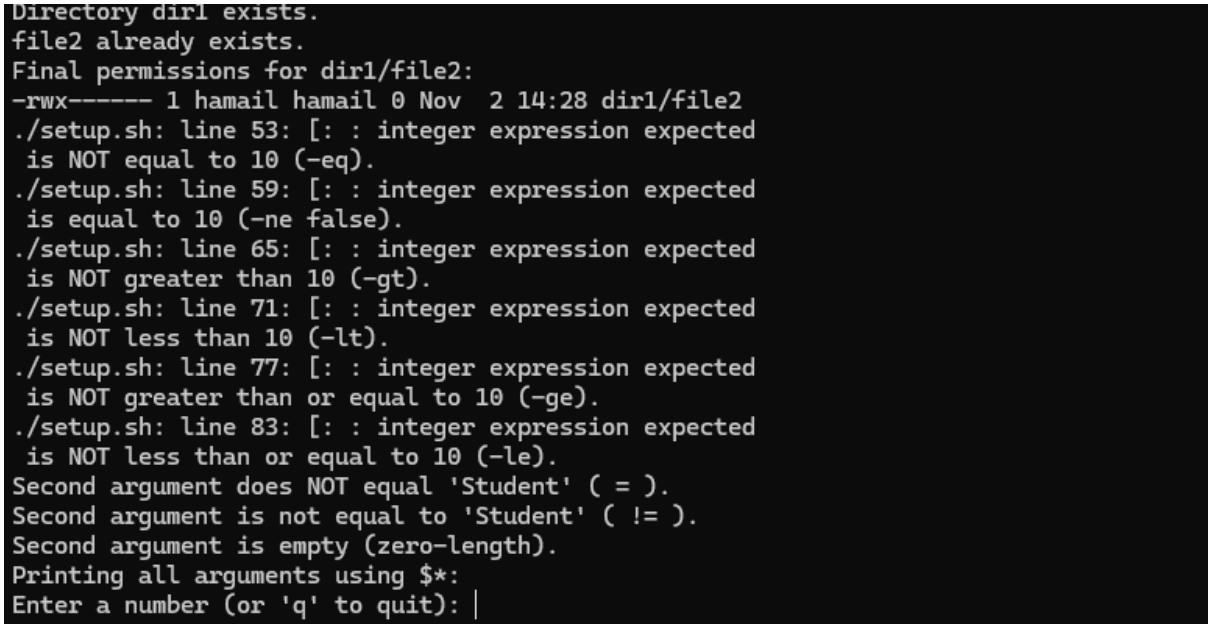
- Screenshots:
  - vim editor showing function appended: task13_b4_vim.png

```
# Demonstrate the function
echo "Now calling sum_two function:"
sum_two

# Function that sums two arguments and returns the result (via exit code)
sum_args_return() {
  a=$1
  b=$2
  return $((a + b))
}

# Demonstrate sum_args_return function
echo "Now demonstrating sum_args_return function:"
sum_args_return 3 4
result=$?
echo "sum_args_return(3,4) returned: $result"
```

  - run output showing function demonstration and returned sum: task13_b4_run.png



Task 14 – Codespaces GUI — fork repo, run start-desktop.sh, open VNC, stop GUI

Goal: Fork the specified repository to your GitHub account, open it in GitHub Codespaces, run the provided script to start a desktop GUI, connect to the GUI via the Codespaces forwarded port (6080) -> vnc.html, and then stop the GUI using the provided stop script.

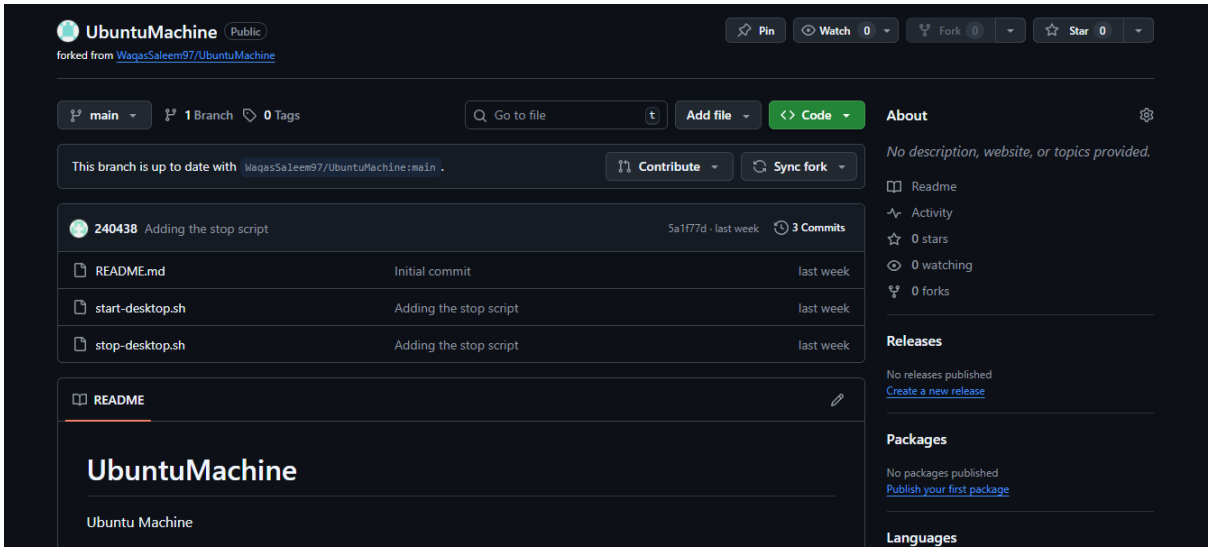
Important notes before starting:

- GitHub Codespaces must be enabled for your account/org. Codespaces availability and billing may apply.
- The instructions below assume you have permission and capacity to create a Codespace for your fork.
- If Codespaces is not available, you may perform this step on another cloud environment that exposes the same port and scripts, but the screenshot filenames below assume Codespaces.

Steps:

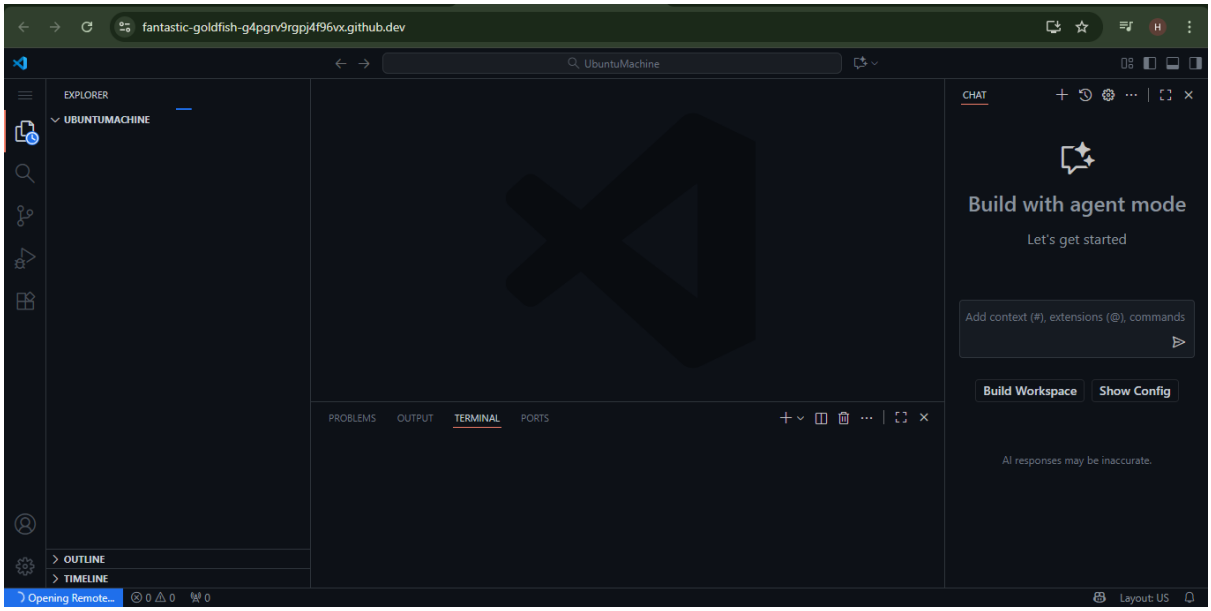
1. Fork the repository to your GitHub account

- Open the repo URL in your browser:
    - Ubuntu Machine
- Click "Fork" (top-right) and fork it to your account.
- Save screenshot as: task14_fork.png



2. Open a Codespace on your fork

- In your forked repository on GitHub, click the green "Code" button → "Open with Codespaces" → "Create codespace on main" (or appropriate branch).
- Wait for the Codespace to initialize.
- Save screenshot as: task14_codespace_launch.png



3. Verify the start script is present and executable (capture evidence)

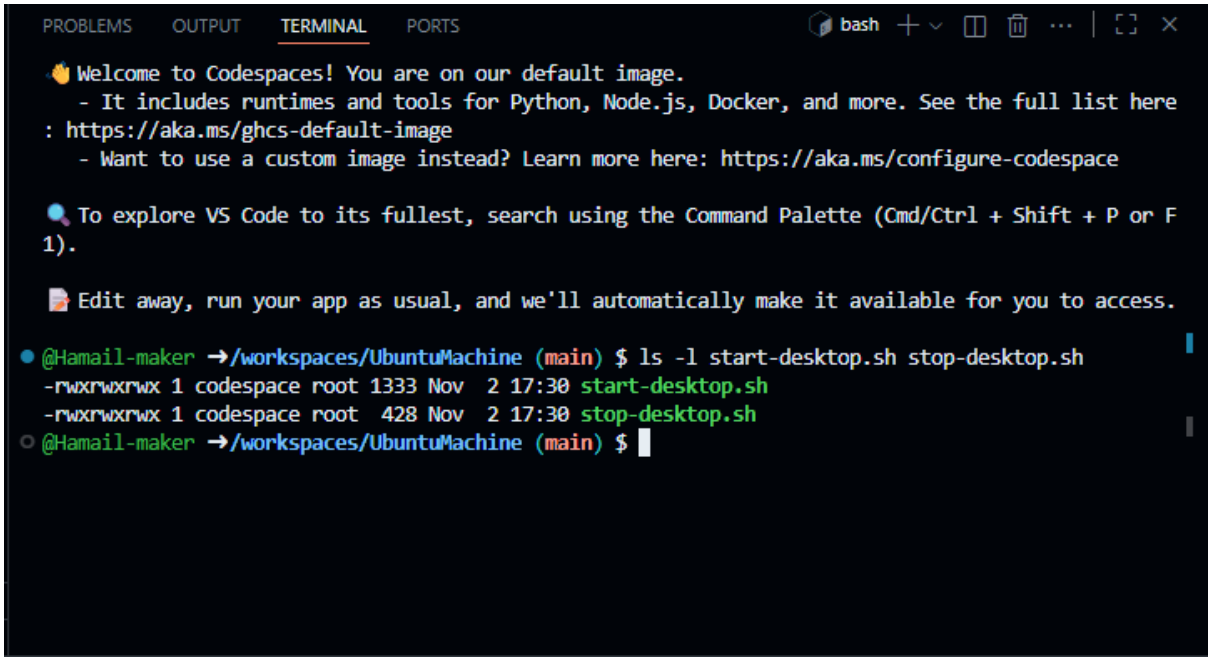
- In the Codespace terminal list files in the repo root and show the start script and stop script exist:

  ls -l start-desktop.sh stop-desktop.sh

  - If not executable, make it executable:

chmod +x start-desktop.sh stop-desktop.sh

- Save a screenshot showing the ls -l output (file listing) and the chmod command if applied:
  - task14_start_script_ls.png



4. Run the start script inside the Codespace terminal
   - In the Codespace terminal run:

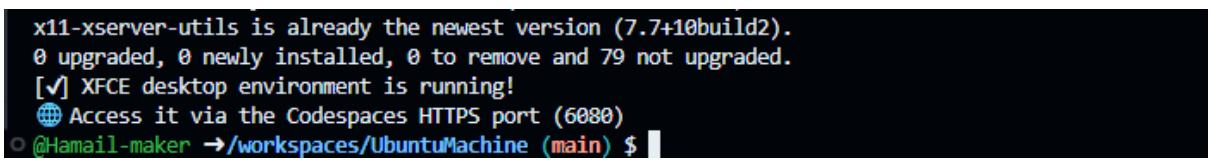
   # Ensure the start script is executable

   chmod +x start-desktop.sh

   # Start the desktop GUI

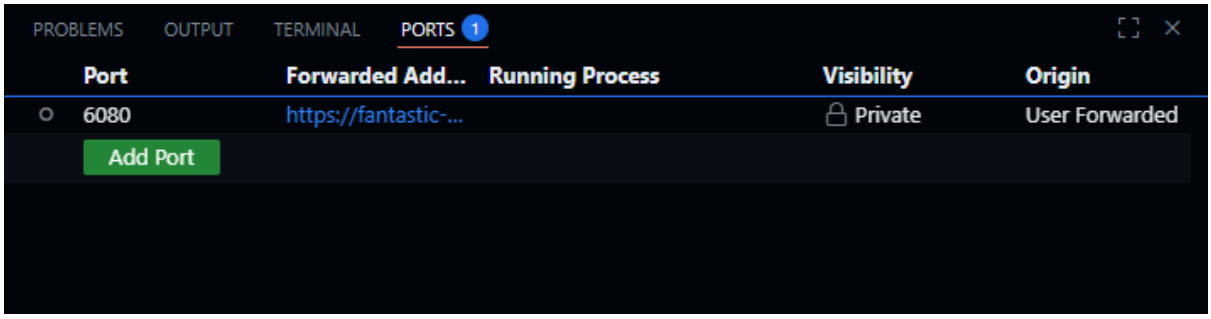
   ./start-desktop.sh

- Capture the terminal output showing successful start messages.
- Save screenshot as: task14_start_run.png



5. Verify forwarded ports in Codespaces (Ports view)

- Open the Codespaces "Ports" panel / view and confirm port 6080 is forwarded and visible.
- Save a screenshot of the Ports view showing port 6080 and its status:
  - task14_ports_view.png



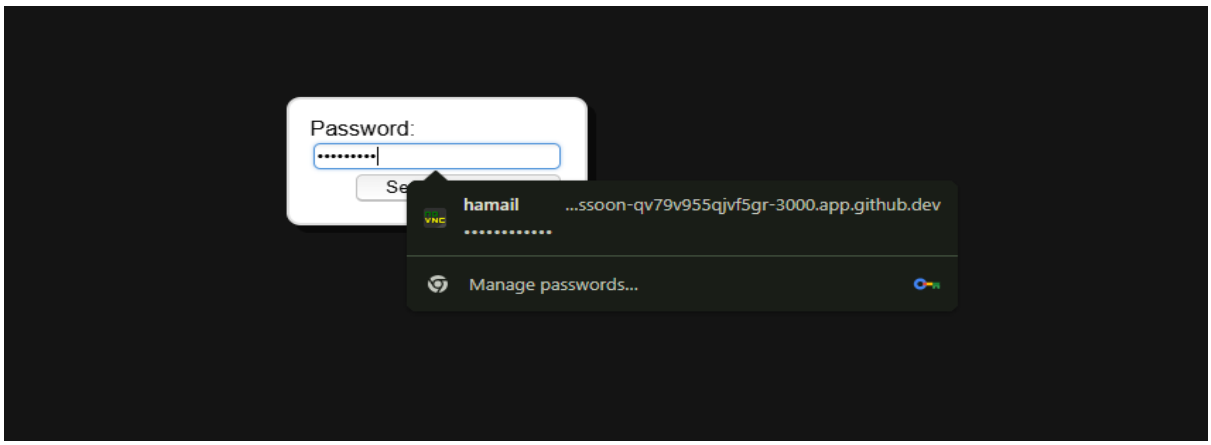
6. Open forwarded port 6080 and connect to VNC HTML page

- In the Codespaces UI, open the forwarded port's preview URL or copy the forwarded URL and open it in your browser.
- Visit the port 6080 address and click the vnc.html link.
- When prompted for a password enter:

  codespace

  - Capture screenshots of:
  - The browser showing the forwarded port URL in the address bar / Codespaces preview: task14_vnc_url.png

  - The VNC password prompt (showing password field; do NOT include typed password in a screenshot): task14_vnc_password_prompt.png



- The VNC session after successful connection showing the GUI/desktop: task14_vnc_desktop.png

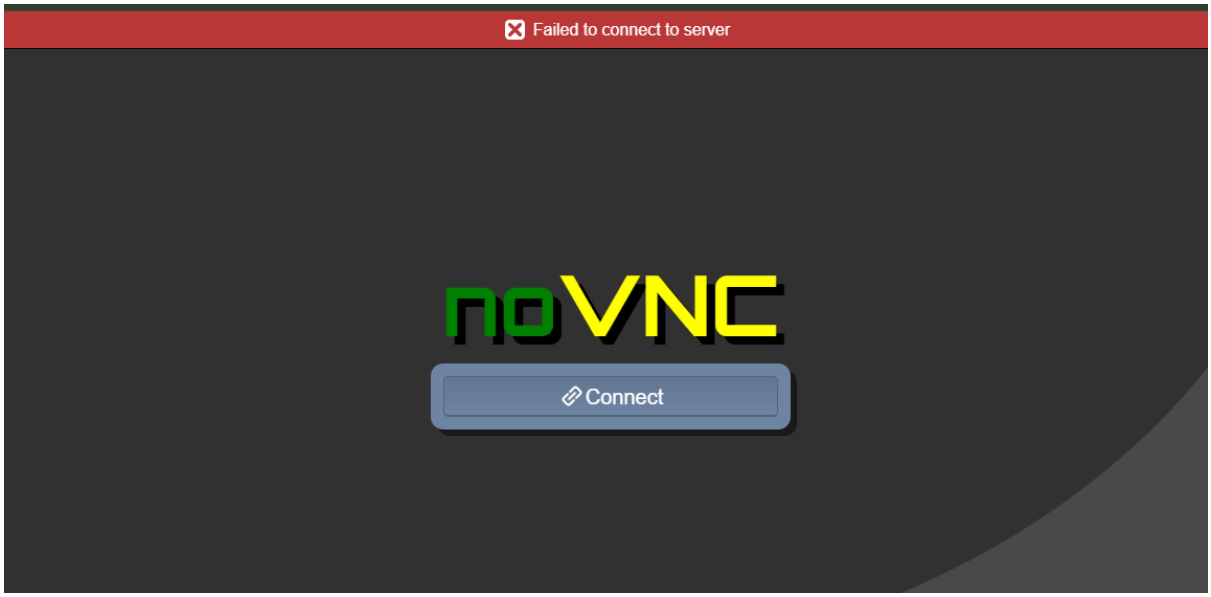
- (Optional) A focused screenshot of vnc.html UI showing the "Connect" button before/after connecting: task14_vnc_connect.png

7. Stop the GUI

- When finished, return to the Codespace terminal and run:

  ./stop-desktop.sh

- Capture the terminal output that shows the GUI stopping and any cleanup messages.
- Save screenshot as: task14_stop_run.png



Exam Evaluation Questions

Use the format below for each exam evaluation question. Each question includes a short scenario and clear numbered steps students must perform; capture the requested screenshots as evidence. These questions cover the concepts taught throughout this lab (users/groups, account files, ownership/permissions, pipes/grep/redirects, scripting

basics, conditionals/comparisons, loops/functions, Codespaces/VNC). Do NOT include answers or solutions in this file.

---

1. Group Management and Membership

Scenario:
Create groups and manage a user's primary and supplementary group memberships.

Steps:

1. Create groups g1, g2, and g3.

- Screenshot: Q1_groups_created.png

```
hamail@ubuntu-lab:~$ sudo groupadd g1
sudo groupadd g2
sudo groupadd g3
[sudo] password for hamail:
hamail@ubuntu-lab:~$
```

1. Change examuser's primary group to g3 and add g1 and g2 as supplementary groups.

- Screenshot: Q1_group_changes.png

```
hamail@ubuntu-lab:~$ sudo usermod -g g3 -aG g1,g2 examuser
usermod: user 'examuser' does not exist
hamail@ubuntu-lab:~$
```

2. Show the final id and /etc/group lines that prove the changes.

- Screenshot: Q1_group_verification.png

```
hamail@ubuntu-lab:~$ id examuser
grep examuser /etc/group
id: ‘examuser’: no such user
hamail@ubuntu-lab:~$
```

---

2. Ownership and Permission Tasks

Scenario:
Demonstrate ownership changes and apply both symbolic and numeric permission changes.

Steps:

1. Create workspace/secret.txt, change its owner to examuser and group to g1.

- Screenshot: Q2_chown_chgrp.png

```
hamail@ubuntu-lab:~$ mkdir ~/workspace
touch ~/workspace/secret.txt
hamail@ubuntu-lab:~$
```

2. Remove all permissions for group and others using a symbolic command, then using a numeric command to achieve the same result.

- Screenshot: Q2_symbolic_numeric.png

```
hamail@ubuntu-lab:~$ sudo chown examuser:g1 ~/workspace/secret.txt
chown: invalid user: ‘examuser:g1’
hamail@ubuntu-lab:~$
```

3. Show ls -l for the file after each change to document the permission bits.

- Screenshot: Q2_permissions_ls.png

```
hamail@ubuntu-lab:~$ chmod go-rwx ~/workspace/secret.txt
ls -l ~/workspace/secret.txt
-rw------- 1 hamail hamail 0 Nov  2 17:56 /home/hamail/workspace/secret.txt
hamail@ubuntu-lab:~$
```

```
hamail@ubuntu-lab:~$ chmod 700 ~/workspace/secret.txt
ls -l ~/workspace/secret.txt
-rwx------ 1 hamail hamail 0 Nov  2 17:56 /home/hamail/workspace/secret.txt
hamail@ubuntu-lab:~$
```

---

3. Pipes, Grep, and Redirection Practice

Scenario:
Filter system logs and save results using redirection and piping.

Steps:

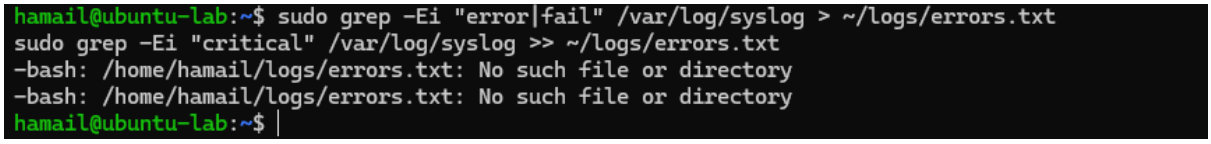
1. Use grep (or journalctl where applicable) with a pipe to find lines containing "error" or "fail" and show the first 20 results.

- Screenshot: Q3_grep_pipe.png

```
hamail@ubuntu-lab:~$ sudo grep -Ei "error|fail" /var/log/syslog | head -20
2025-11-02T10:02:19.414360+00:00 ubuntu-lab multipathd[494]: sda: failed to get udev uid: No data available
2025-11-02T10:02:19.420626+00:00 ubuntu-lab multipath: sda: failed to get sysfs uid: No such file or directory
2025-11-02T10:02:19.420642+00:00 ubuntu-lab multipath: sda: failed to get sgio uid: No such file or directory
2025-11-02T10:02:19.420660+00:00 ubuntu-lab multipathd[494]: sda: failed to get udev uid: No data available
2025-11-02T10:02:19.420665+00:00 ubuntu-lab multipathd[494]: sda: failed to get path uid
2025-11-02T10:02:19.420971+00:00 ubuntu-lab multipathd[494]: uevent trigger error
2025-11-02T10:02:19.425920+00:00 ubuntu-lab kernel: ACPI: _OSC evaluation for CPUs failed, trying _PDC
2025-11-02T10:02:19.426610+00:00 ubuntu-lab systemd[1]: apport-autoreport.path - Process error reports when automatic re
porting is enabled (file watch) was skipped because of an unmet condition check (ConditionPathExists=/var/lib/apport/aut
oreport).
2025-11-02T10:02:19.426625+00:00 ubuntu-lab systemd[1]: apport-autoreport.timer - Process error reports when automatic r
eporting is enabled (timer based) was skipped because of an unmet condition check (ConditionPathExists=/var/lib/apport/a
utoreport).
2025-11-02T10:02:19.435887+00:00 ubuntu-lab systemd[1]: rsyslog.service: Failed with result 'signal'.
2025-11-02T10:02:19.435897+00:00 ubuntu-lab systemd[1]: Failed to start rsyslog.service - System Logging Service.
2025-11-02T10:02:19.436746+00:00 ubuntu-lab systemd[1]: Starting grub-initrd-fallback.service - GRUB failed boot detecti
on...
2025-11-02T10:02:19.437216+00:00 ubuntu-lab systemd[1]: Finished grub-initrd-fallback.service - GRUB failed boot detecti
on.
2025-11-02T10:02:19.437964+00:00 ubuntu-lab kernel: pci 0000:00:15.3: bridge window [io  size 0x1000]: failed to assign
2025-11-02T10:02:19.437965+00:00 ubuntu-lab kernel: pci 0000:00:15.4: bridge window [io  size 0x1000]: failed to assign
2025-11-02T10:02:19.437965+00:00 ubuntu-lab kernel: pci 0000:00:15.5: bridge window [io  size 0x1000]: failed to assign
2025-11-02T10:02:19.437966+00:00 ubuntu-lab kernel: pci 0000:00:15.6: bridge window [io  size 0x1000]: failed to assign
2025-11-02T10:02:19.437967+00:00 ubuntu-lab kernel: pci 0000:00:15.7: bridge window [io  size 0x1000]: failed to assign
2025-11-02T10:02:19.437968+00:00 ubuntu-lab kernel: pci 0000:00:16.3: bridge window [io  size 0x1000]: failed to assign
2025-11-02T10:02:19.437969+00:00 ubuntu-lab kernel: pci 0000:00:16.4: bridge window [io  size 0x1000]: failed to assign
hamail@ubuntu-lab:~$
```

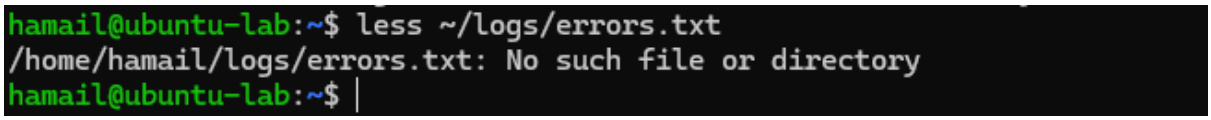
2. Save the filtered results to a file ~/logs/errors.txt using overwrite, then append additional matching lines using append redirection.

- Screenshot: Q3_redirect_overwrite_append.png



3. Use a pager to view the saved file.

- Screenshot: Q3_pager_view.png



---

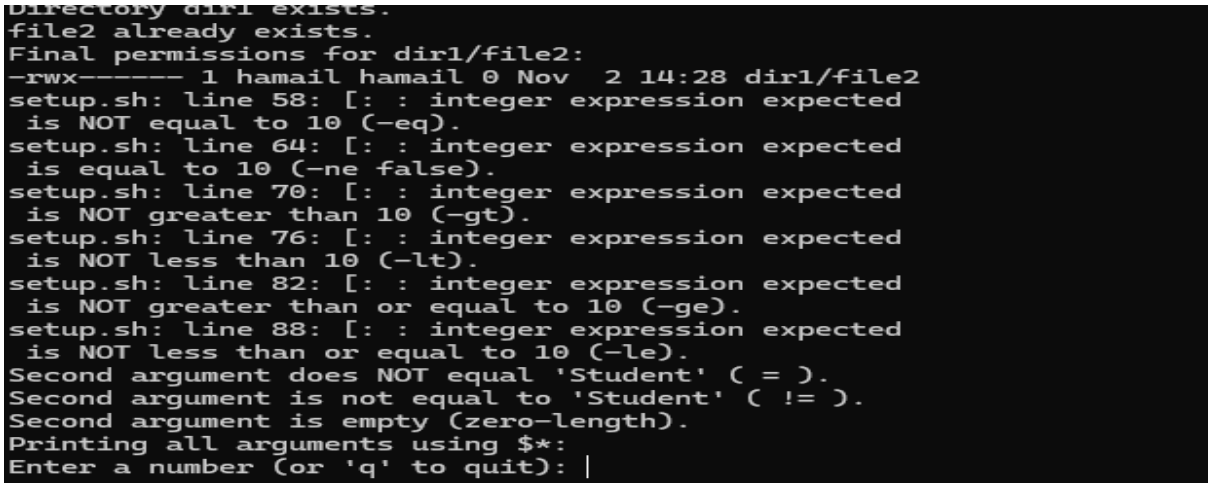
4. Script: Variables, Command Substitution, File & Dir Checks

Scenario:
Build and run a script incrementally that demonstrates variables, command substitution, and filesystem checks.

Steps:

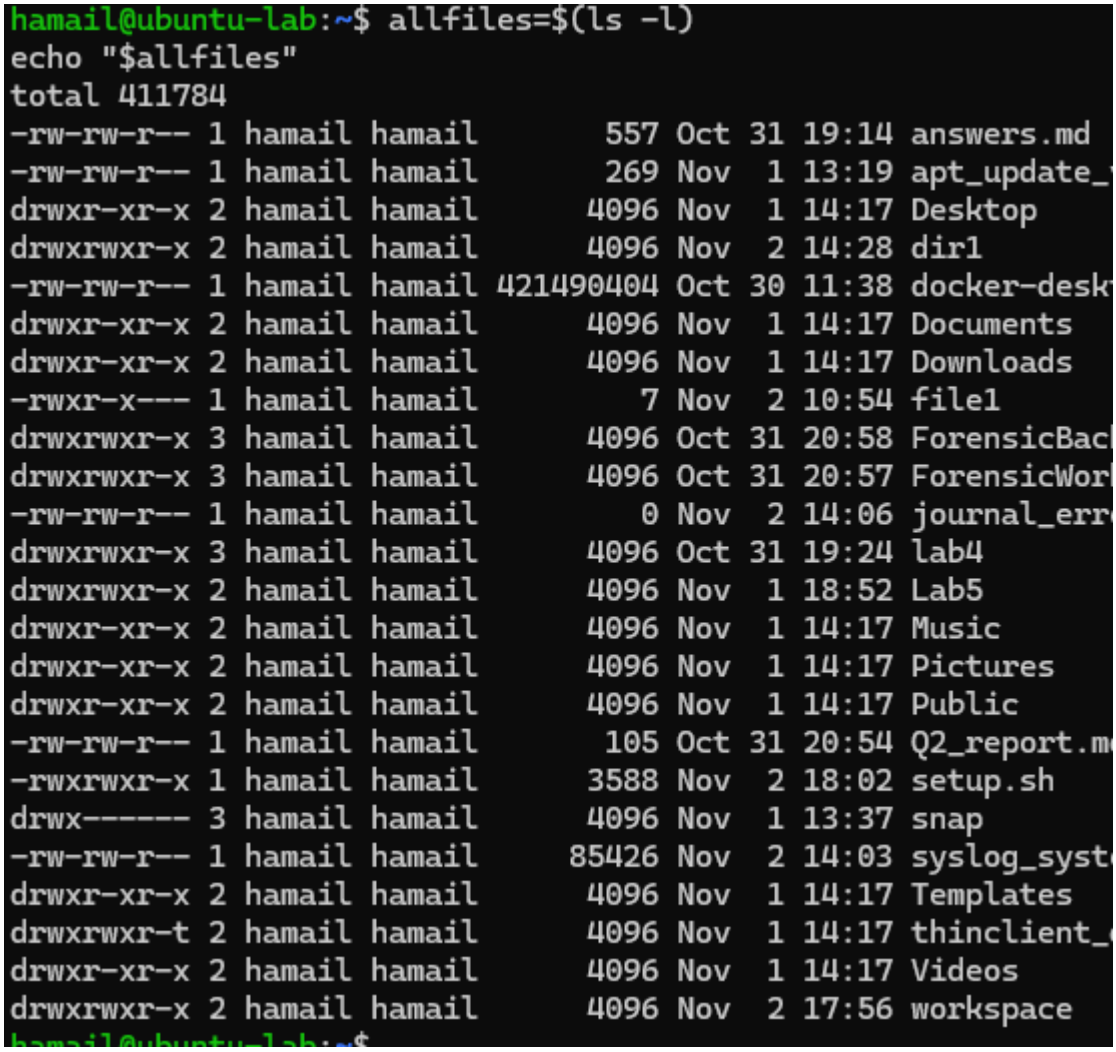
1. Create setup.sh with a shebang and a variable var1 that you echo.

- Screenshot: Q4_step1_var1.png



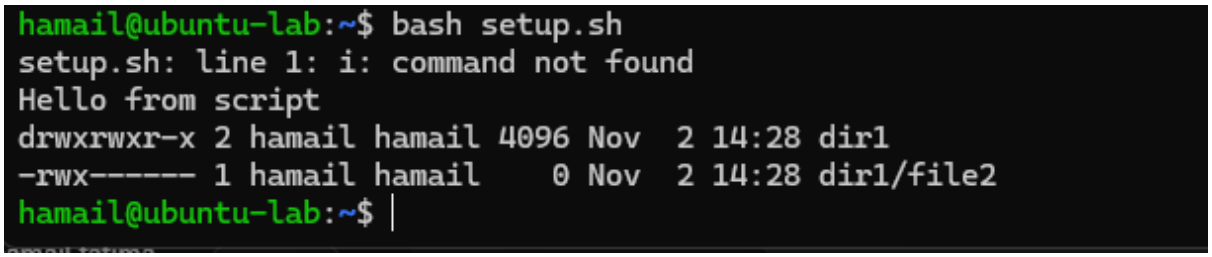
2. Append command substitution that stores ls -l output into a variable and echo it.

- Screenshot: Q4_step2_allfiles.png



3. Append directory and file checks that create dir1 and dir1/file2 if missing, and display their final permissions.

- Screenshot: Q4_step3_dirfile_checks.png



---

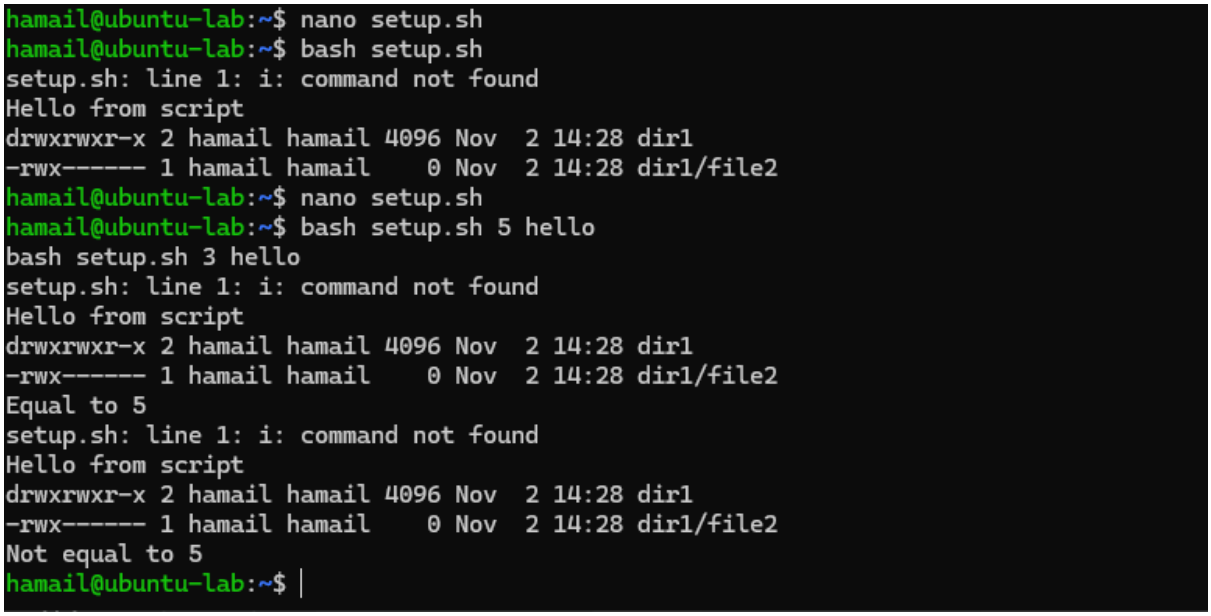
5. Script: Comparisons and String Tests

Scenario:
Incrementally add numeric and string comparison tests to a script and show both true/false cases.

Steps:

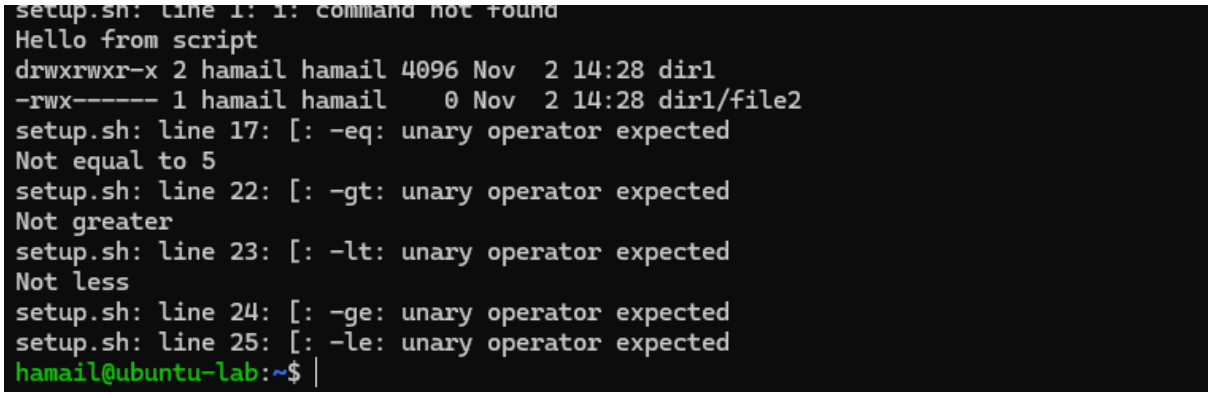
1. Overwrite setup.sh to set num=$1 and str=$2, and add an -eq test showing true and false examples.

- Screenshot: Q5_eq_examples.png



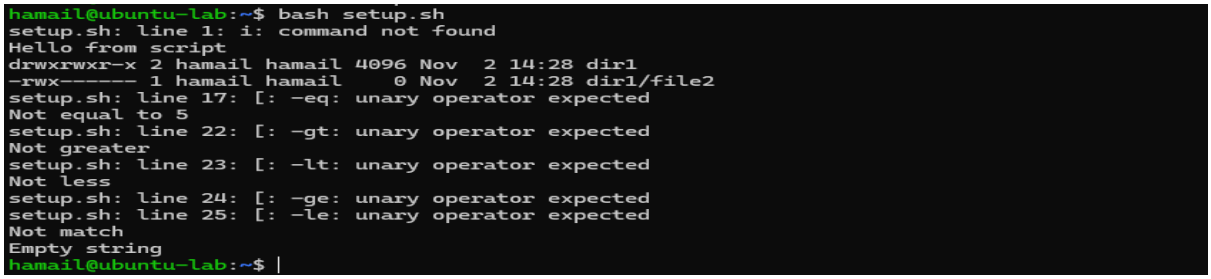
2. Append -ne, -gt, -lt, -ge, and -le tests and demonstrate at least one true and one false invocation for each.

- Screenshot: Q5_numeric_tests.png



3. Append string equality (=) and inequality (!=) checks and a -z (zero-length) test for the second argument, demonstrating true/false cases.

- Screenshot: Q5_string_tests.png



---

6. Script: For Loop and Argument Handling

Scenario:
Write a script that prints all provided arguments and demonstrate correct handling of quoted multi-word arguments.

Steps:

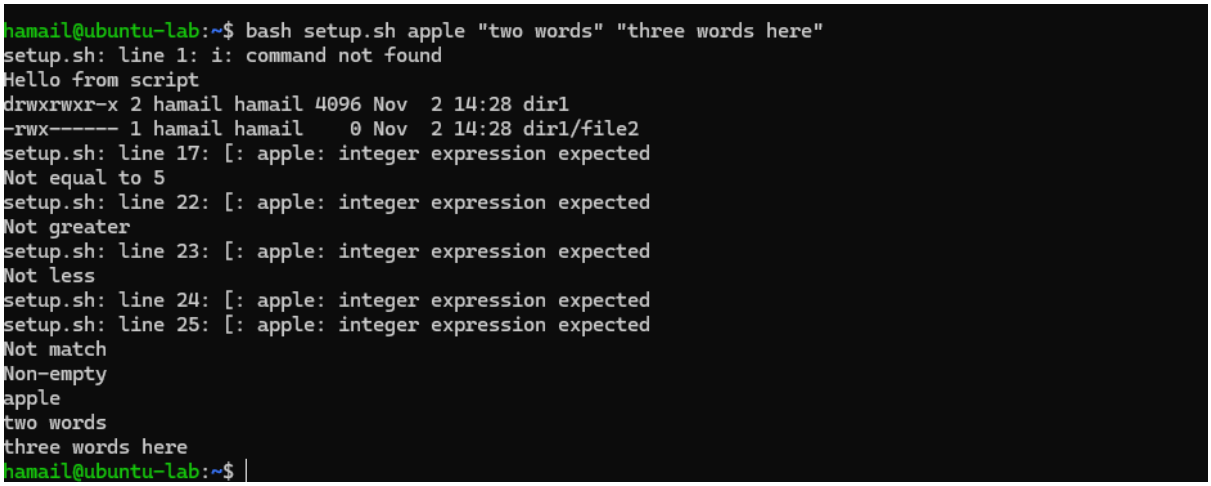
1. Create/overwrite setup.sh to print every argument using "$@" in a for loop and save the file.

- Screenshot: Q6_script_forloop_vim.png

```
#!/bin/bash
for arg in "$@"
do
  echo "$arg"
done
```

2. Run the script with mixed single and quoted multi-word arguments and capture the output showing each argument on its own line.

- Screenshot: Q6_forloop_run.png



```
hamail@ubuntu-lab:~$ bash setup.sh apple "two words" "three words here"
setup.sh: line 1: i: command not found
Hello from script
drwxrwxr-x 2 hamail hamail 4096 Nov  2 14:28 dir1
-rwx------ 1 hamail hamail    0 Nov  2 14:28 dir1/file2
setup.sh: line 17: [: apple: integer expression expected
Not equal to 5
setup.sh: line 22: [: apple: integer expression expected
Not greater
setup.sh: line 23: [: apple: integer expression expected
Not less
setup.sh: line 24: [: apple: integer expression expected
setup.sh: line 25: [: apple: integer expression expected
Not match
Non-empty
apple
two words
three words here
hamail@ubuntu-lab:~$
```

---

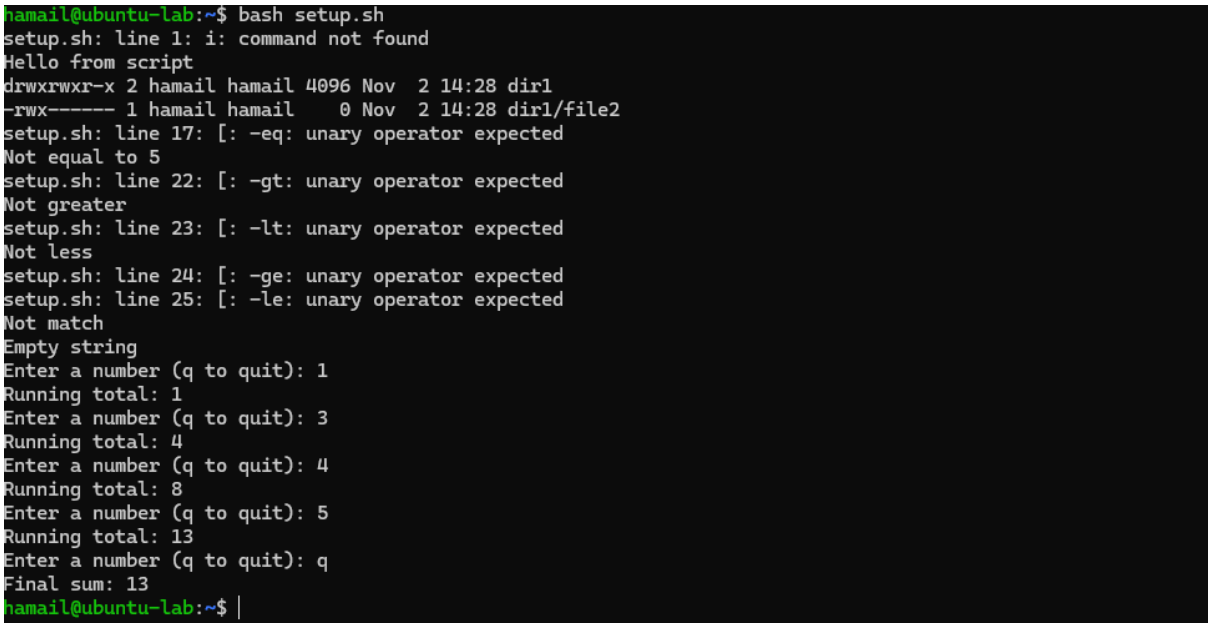
7. Script: While Loop Summation and Functions

Scenario:
Implement an interactive or non-interactive summation function and a demonstrated function that returns a numeric result.

Steps:

1. Write an interactive while-loop that accumulates numbers until q is entered and shows running totals.

- Screenshot: Q7_while_session.png



2. Add a function that accepts two numeric arguments, returns their sum, and demonstrate capturing its result in a variable.

- Screenshot: Q7_function_sum.png