Name: Lujain Zia
Roll no: 2023-BSE-034
Date: October 31, 2025

Lab 6

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

```
ubuntu@ubuntuserver:~$ sudo passwd root
[sudo] password for ubuntu:
Sorry, try again.
[sudo] password for ubuntu:
New password:
Retype new password:
passwd: password updated successfully
ubuntu@ubuntuserver:~$
```

2. Switch to root and verify:

su -
whoami
id

Save screenshot as: task1_su_root.png

```
ubuntu@ubuntuserver:~$ su -
Password:
root@ubuntuserver:~# whoami
root
root@ubuntuserver:~# id
uid=0(root) gid=0(root) groups=0(root)
root@ubuntuserver:~#
```

3. Switch back to your normal user:

exit
whoami

Save screenshot as: task1_exit_to_user.png

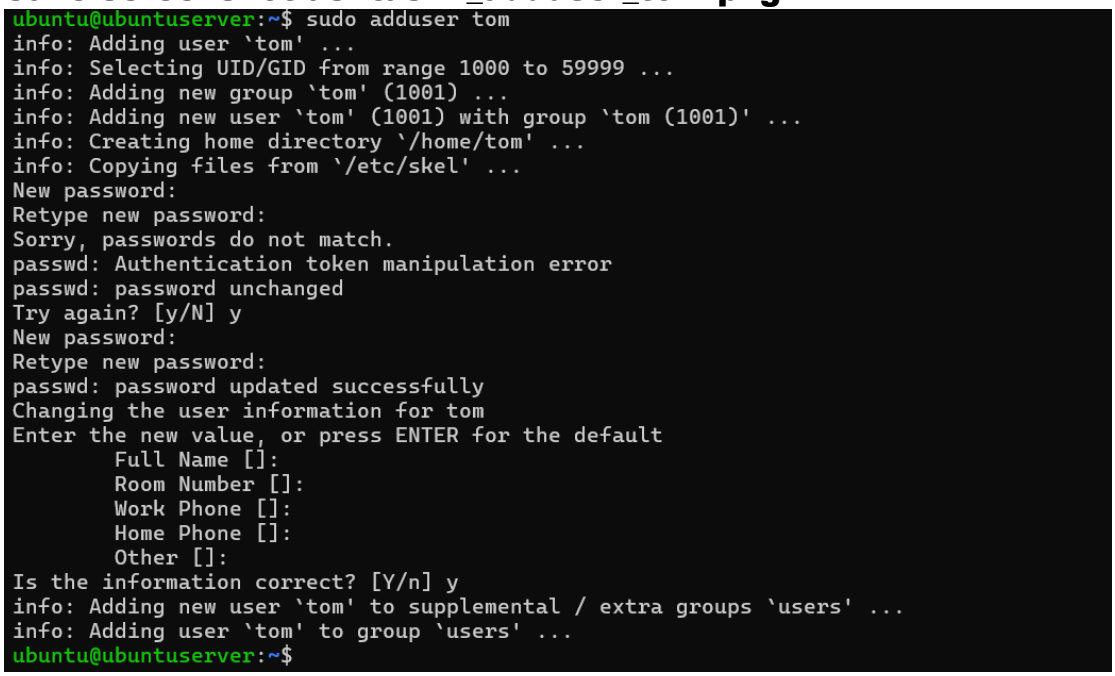
```
root@ubuntuserver:~# exit
logout
ubuntu@ubuntuserver:~$ whoami
ubuntu
ubuntu@ubuntuserver:~$
```

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

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

sudo adduser tom

Save screenshot as: task2_adduser_tom.png



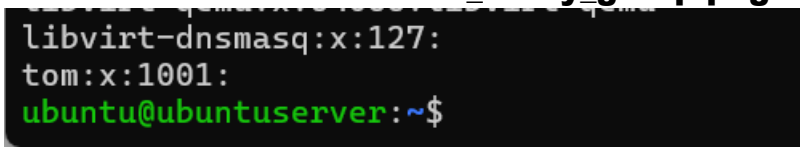
2. Verify tom in system files (view and visually confirm presence):

cat /etc/passwd

Save screenshot as: task2_verify_passwd.png

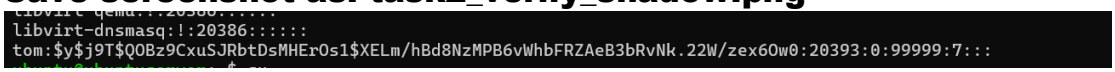
cat /etc/group

Save screenshot as: task2_verify_group.png



sudo cat /etc/shadow

Save screenshot as: task2_verify_shadow.png



Notes:

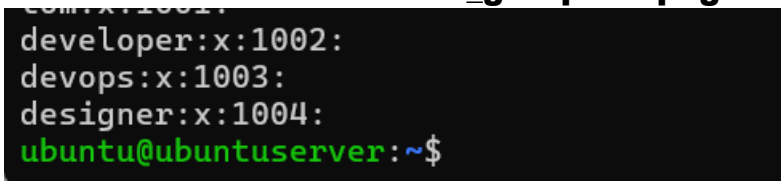
/etc/shadow stores password hashes (not plaintext). You must use sudo to read it.

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

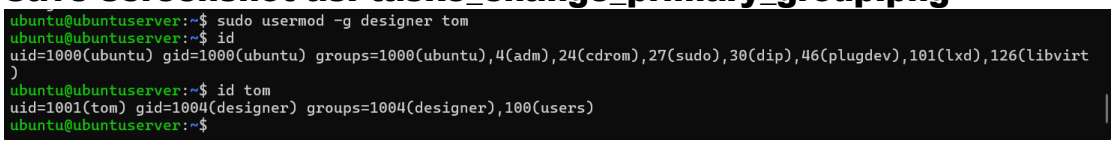
Save screenshot as: task3_groupadd.png



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

sudo usermod -g designer tom
id tom

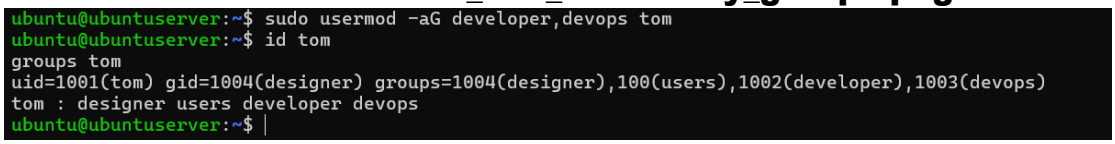
Save screenshot as: task3_change_primary_group.png



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

sudo usermod -aG developer,devops tom
id tom
groups tom

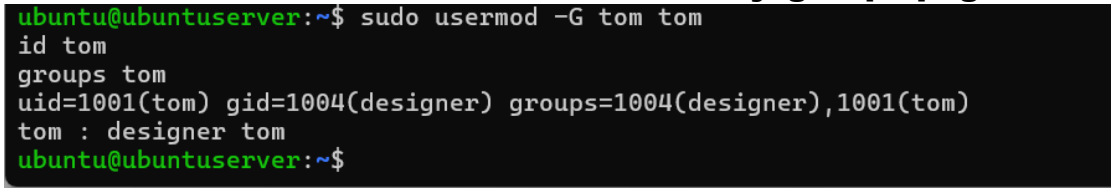
Save screenshot as: task3_add_secondary_groups.png



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)

1. Create users:

sudo adduser Jerry

sudo useradd Scooby

Save screenshot as: task4_add_users.png

```
ubuntu@ubuntuserver:~$ sudo useradd scooby
ubuntu@ubuntuserver:~$ sudo adduser jerry
info: Adding user `jerry' ...
info: Selecting UID/GID from range 1000 to 59999 ...
info: Adding new group `jerry' (1007) ...
info: Adding new user `jerry' (1007) with group `jerry (1007)' ...
info: Creating home directory `/home/jerry' ...
info: Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for jerry
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
info: Adding new user `jerry' to supplemental / extra groups `users' ...
info: Adding user `jerry' to group `users' ...
ubuntu@ubuntuserver:~$
```

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

```
ubuntu@ubuntuserver:~$ su - scooby
Password:
su: Authentication failure
ubuntu@ubuntuserver:~$
```

3. Set a password for Scooby:

sudo passwd Scooby

Save screenshot as: task4_set_password_scooby.png

```
ubuntu@ubuntuserver:~$ sudo passwd Scooby
New password:
Retype new password:
passwd: password updated successfully
ubuntu@ubuntuserver:~$
```

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

```
ubuntu@ubuntuserver:~$ su -scooby
Password:
su: failed to execute cooby: No such file or directory
ubuntu@ubuntuserver:~$
```

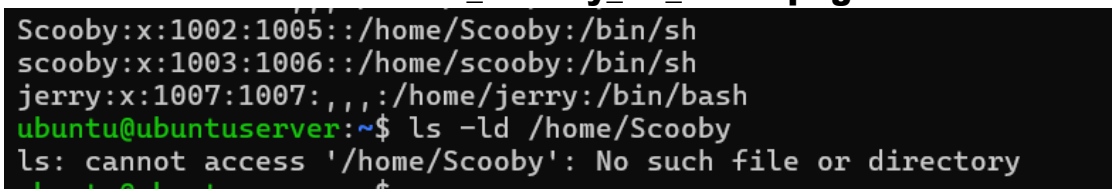
5. Show that Scooby’s home directory does not exist yet and what /etc/passwd says:

exit

cat /etc/passwd

ls -ld /home/Scooby

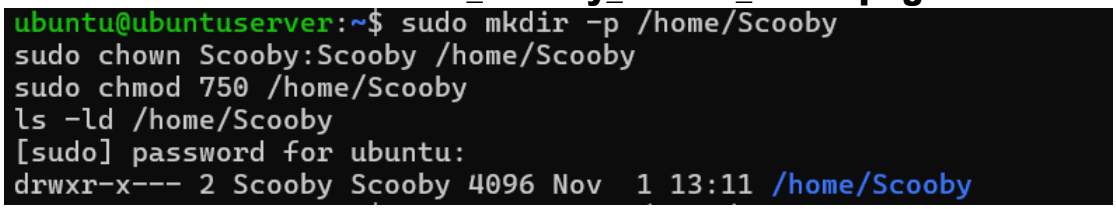
Save screenshot as: task4_scooby_no_home.png



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

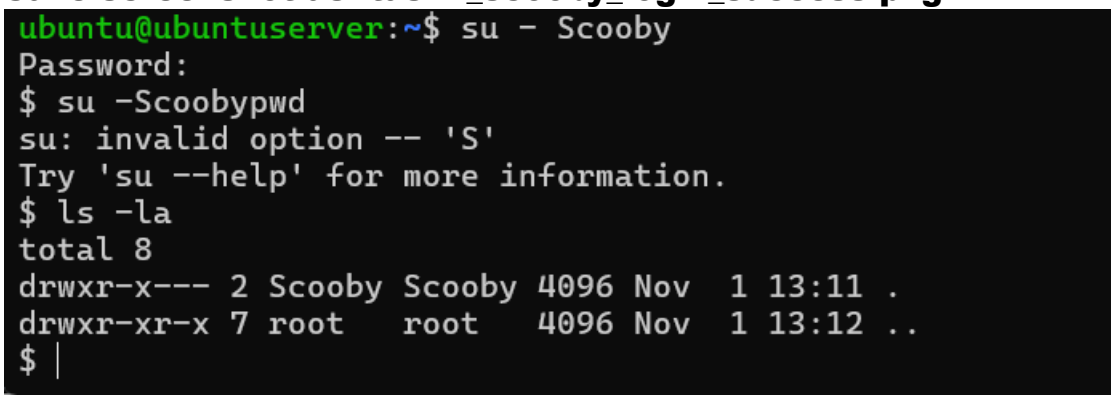
Save screenshot as: task4_scooby_create_home.png



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

su - Scoobypwd
ls -la

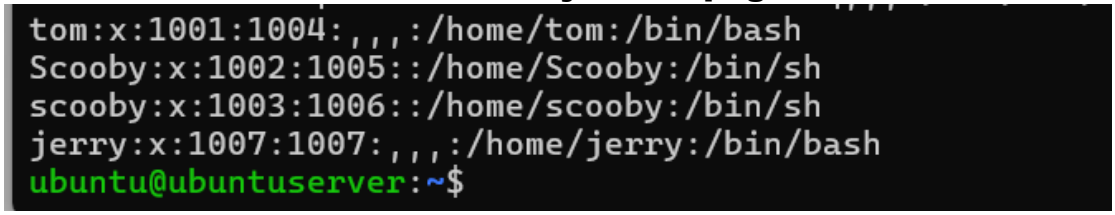
Save screenshot as: task4_scooby_login_success.png



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

```
ubuntu@ubuntuserver:~$ sudo usermod -s /bin/bash Scooby
su - Scooby
Password:
Scooby@ubuntuserver:~$ exit
logout
tom:x:1001:1004:,,,:/home/tom:/bin/bash
Scooby:x:1002:1005::/home/Scooby:/bin/bash
scooby:x:1003:1006::/home/scooby:/bin/sh
jerry:x:1007:1007:,,,:/home/jerry:/bin/bash
ubuntu@ubuntuserver:~$
```

10. Create groups:

sudo addgroup jolly
sudo groupadd anime

Save screenshot as: task4_add_groups.png

```
ubuntu@ubuntuserver:~$ sudo addgroup jolly
sudo groupadd anime
info: Selecting GID from range 1000 to 59999 ...
info: Adding group `jolly' (GID 1008) ...
ubuntu@ubuntuserver:~$
```

11. Verify groups:

cat /etc/group

Save screenshot as: task4_verify_groups.png

```
jerry:x:1007:
jolly:x:1008:
anime:x:1009:
ubuntu@ubuntuserver:~$
```

12. Delete groups and users:

sudo delgroup jolly
sudo groupdel anime
cat /etc/group

sudo deluser --remove-home Jerry
sudo userdel -r Scooby
cat /etc/passwd

Save screenshots

as: task4_delete_groups.png, task4_delete_users.png

Deleting groups:

```
ubuntu@ubuntuserver:~$ sudo delgroup jolly
sudo groupdel anime
cat /etc/group
info: Removing group `jolly' ...
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog,ubuntu
```

```
designer:x:1004:
Scooby:x:1005:
scooby:x:1006:
jerry:x:1007:
ubuntu@ubuntuserver:~$
```

Deleting users:

```
ubuntu@ubuntuserver:~$ sudo deluser --remove-home jerry
sudo userdel -r Scooby
cat /etc/passwd
info: Looking for files to backup/remove ...
info: Removing files ...
info: Removing crontab ...
info: Removing user `jerry' ...
userdel: user 'Scooby' does not exist
libvirt-qemu:x:64055:994:Libvirt Qemu,,,:/var/lib/libvi
libvirt-dnsmasq:x:121:127:Libvirt Dnsmasq,,,:/var/lib/li
tom:x:1001:1004:,,,:/home/tom:/bin/bash
```

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

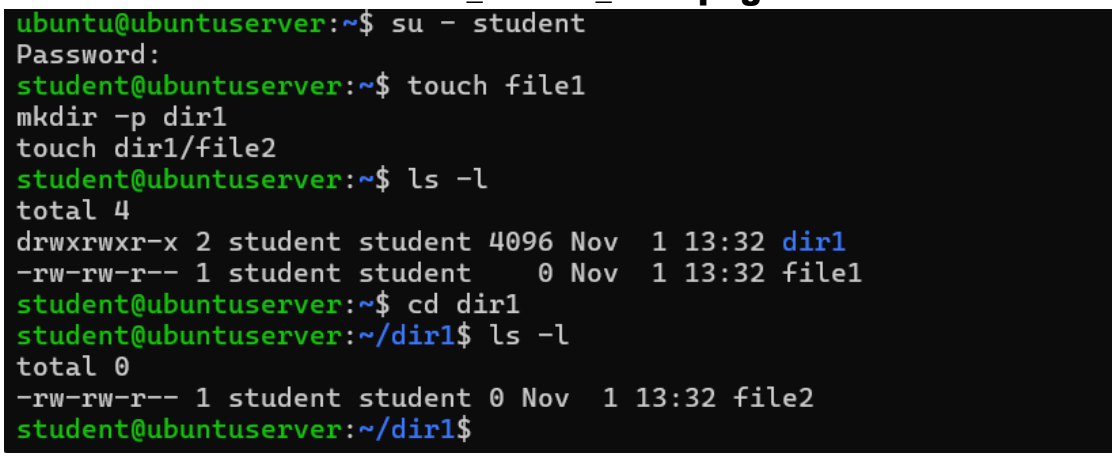
```
ubuntu@ubuntuserver:~$ sudo adduser student
info: Adding user `student' ...
info: Selecting UID/GID from range 1000 to 59999 ...
info: Adding new group `student' (1005) ...
info: Adding new user `student' (1005) with group `student (1005)' ...
info: Creating home directory `/home/student' ...
info: Copying files from `/etc/skel' ...
New password:
Retype new password:
Sorry, passwords do not match.
passwd: Authentication token manipulation error
passwd: password unchanged
Try again? [y/N] y
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for student
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
info: Adding new user `student' to supplemental / extra groups `users' ...
info: Adding user `student' to group `users' ...
ubuntu@ubuntuserver:~$
```

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

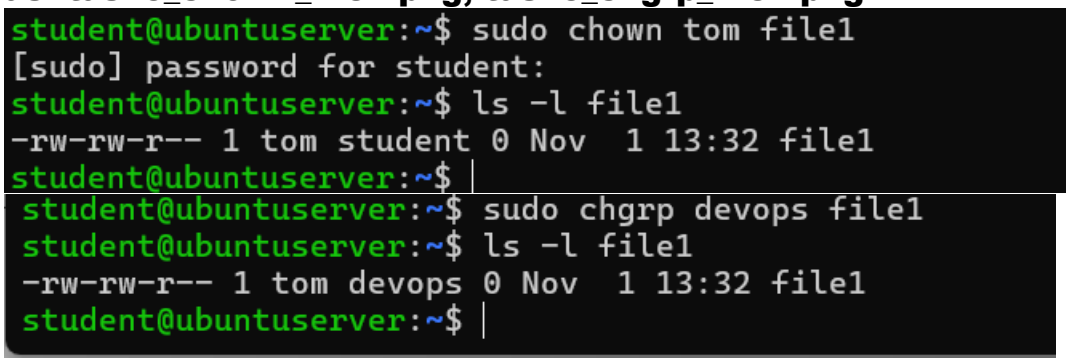


3. Change owner then group for file1 (separate commands):

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

```
student@ubuntuserver:~$ ls -l
ls -l dir1
total 4
drwxrwxr-x 2 student student 4096 Nov  1 13:32 dir1
-rw-rw-r-- 1 tom     devops     0 Nov  1 13:32 file1
total 0
-rw-rw-r-- 1 student student 0 Nov  1 13:32 file2
student@ubuntuserver:~$ ls -l /dev/null
file file1 dir1 /dev/null
crw-rw-rw- 1 root root 1, 3 Nov  1 12:00 /dev/null
file1:     empty
dir1:      directory
/dev/null: character special (1/3)
student@ubuntuserver:~$
```

5. Exit Student:

exit

Save screenshot as: task5_exit_student.png

```
student@ubuntuserver:~$ exit
logout
ubuntu@ubuntuserver:~$
```

Task 6 – Change permissions using symbolic mode

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

1. Ensure Student and file present:

su - Studentcd ~

ls -l file1

Save screenshot as: task6_su_student.png

```
student@ubuntuserver:~$ ls -l file1
-rw-rw-r-- 1 tom devops 0 Nov  1 13:32 file1
student@ubuntuserver:~$
```

2. Remove all permissions:

chmod -rwx file1

ls -l file1

Save screenshot as: task6_chmod_remove_rwx.png

```
student@ubuntuserver:~$ chmod -rwx file1
student@ubuntuserver:~$ ls -l file1
---------- 1 student devops 0 Nov  1 13:32 file1
student@ubuntuserver:~$
```

3. Add read to all:

chmod +r file1

ls -l file1

Save screenshot as: task6_chmod_add_r.png

```
student@ubuntuserver:~$ chmod +r file1
ls -l file1
-r--r--r-- 1 student devops 0 Nov  1 13:32 file1
student@ubuntuserver:~$
```

4. Add execute to user:
chmod u+x file1
ls -l file1
Save screenshot as: task6_chmod_u_plus_x.png

```
student@ubuntuserver:~$ chmod u+x file1
ls -l file1
-r-xr--r-- 1 student devops 0 Nov  1 13:32 file1
student@ubuntuserver:~$
```

5. Add write to user and group:
chmod ug+w file1
ls -l file1
Save screenshot as: task6_chmod_ug_plus_w.png

```
student@ubuntuserver:~$ chmod ug+w file1
ls -l file1
-rwxrw-r-- 1 student devops 0 Nov  1 13:32 file1
student@ubuntuserver:~$
```

6. Remove all permissions (explicit):
chmod ugo-rwx file1
ls -l file1
Save screenshot as: task6_chmod_ugo_minus_rwx.png

```
student@ubuntuserver:~$ chmod ugo-rwx file1
ls -l file1
---------- 1 student devops 0 Nov  1 13:32 file1
```

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

1. Ensure you are Student:
su - Studentcd ~
ls -l file1
Save screenshot as: task7_student_context.png

```
student@ubuntuserver:~$ ls -l file1
---------- 1 student devops 0 Nov  1 13:32 file1
student@ubuntuserver:~$
```

2. Set all to rwx:
chmod u=rwx,g=rwx,o=rwx file1
ls -l file1
Save screenshot as: task7_chmod_set_all_rwx.png

```
student@ubuntuserver:~$ chmod u=rwx,g=rwx,o=rwx file1
ls -l file1
-rwxrwxrwx 1 student devops 0 Nov  1 13:32 file1
student@ubuntuserver:~$
```

3. Remove execute from group and others:
chmod g=rw,o=rw file1
ls -l file1
Save screenshot as: task7_remove_exec_go.png

```
student@ubuntuserver:~$ chmod g=rw,o=rw file1
ls -l file1
-rwxrw-rw- 1 student devops 0 Nov  1 13:32 file1
student@ubuntuserver:~$
```

4. Remove all permissions:

chmod u=,g=,o= file1
ls -l file1

Save screenshot as: task7_remove_all_perms.png

```
student@ubuntuserver:~$ chmod u=,g=,o= file1
ls -l file1
---------- 1 student devops 0 Nov  1 13:32 file1
student@ubuntuserver:~$
```

Task 8 – Change permissions using numeric (octal) mode

1. Ensure you are Student:

su - Studentcd ~
ls -l file1

Save screenshot as: task8_student_context.png

```
student@ubuntuserver:~$ ls -l file1
---------- 1 student devops 0 Nov  1 13:32 file1
student@ubuntuserver:~$
```

2.Run each command and capture screenshot after each ls:

chmod 777 file1
ls -l file1

task8_chmod_777.png

```
student@ubuntuserver:~$ chmod 777 file1
ls -l file1
-rwxrwxrwx 1 student devops 0 Nov  1 13:32 file1
student@ubuntuserver:~$
```

3.

chmod 700 file1
ls -l file1

task8_chmod_700.png

```
student@ubuntuserver:~$ chmod 700 file1
ls -l file1
-rwx------ 1 student devops 0 Nov  1 13:32 file1
student@ubuntuserver:~$
```

4.

chmod 744 file1
ls -l file1

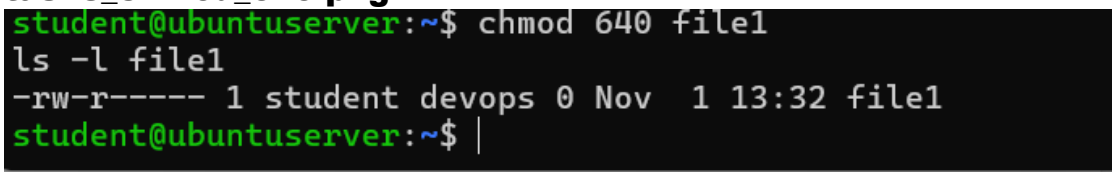
task8_chmod_744.png

```
student@ubuntuserver:~$ chmod 744 file1
ls -l file1
-rwxr--r-- 1 student devops 0 Nov  1 13:32 file1
student@ubuntuserver:~$
```

5.

chmod 640 file1
ls -l file1

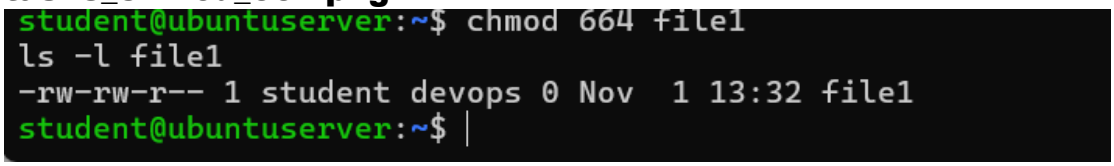
task8_chmod_640.png



6.

chmod 664 file1
ls -l file1

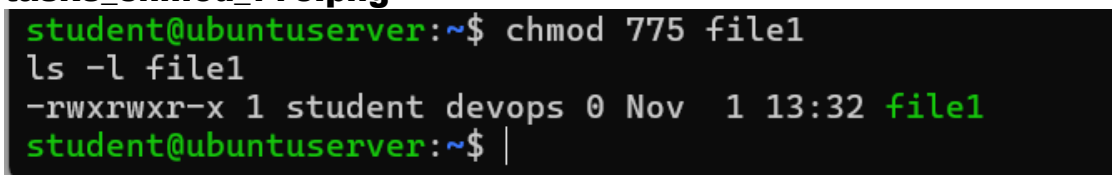
task8_chmod_664.png



7.

chmod 775 file1
ls -l file1

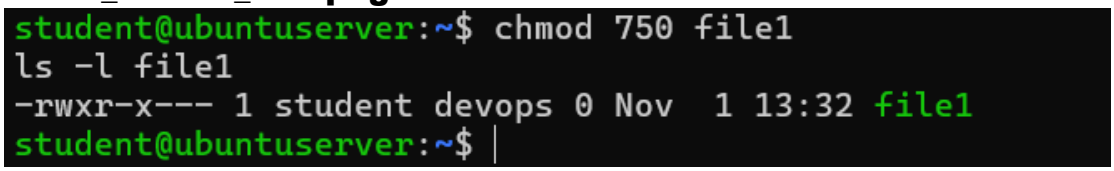
task8_chmod_775.png



8.

chmod 750 file1
ls -l file1

task8_chmod_750.png

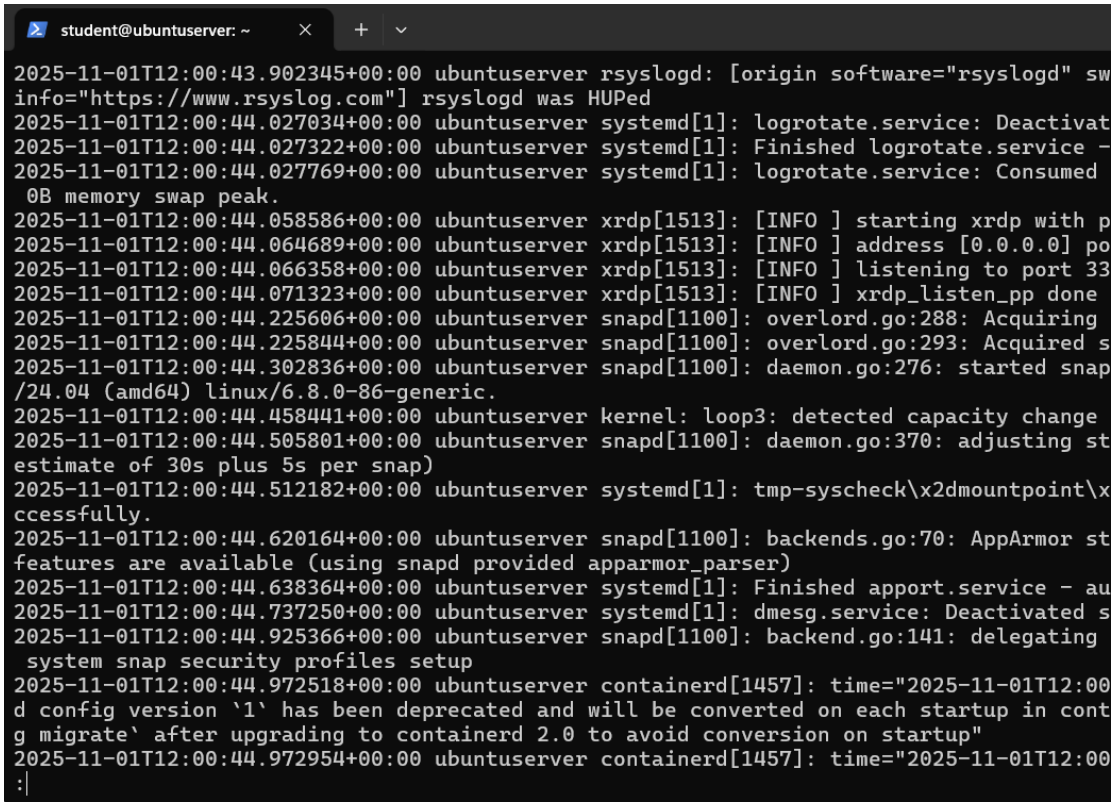


Task 9 – Practice pipes, pagers, grep, and redirects with /var/log/syslog

1. less:

sudo cat /var/log/syslog | less # quit q

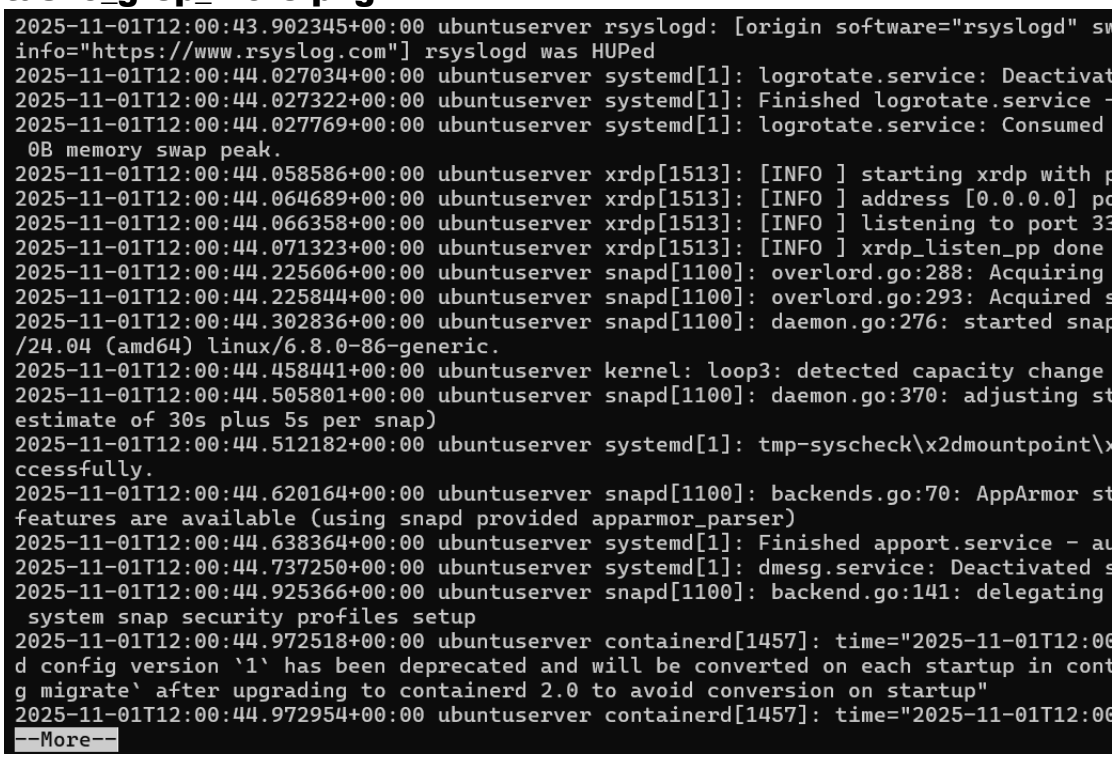
task9_grep_less.png



2. more:

sudo cat /var/log/syslog | more

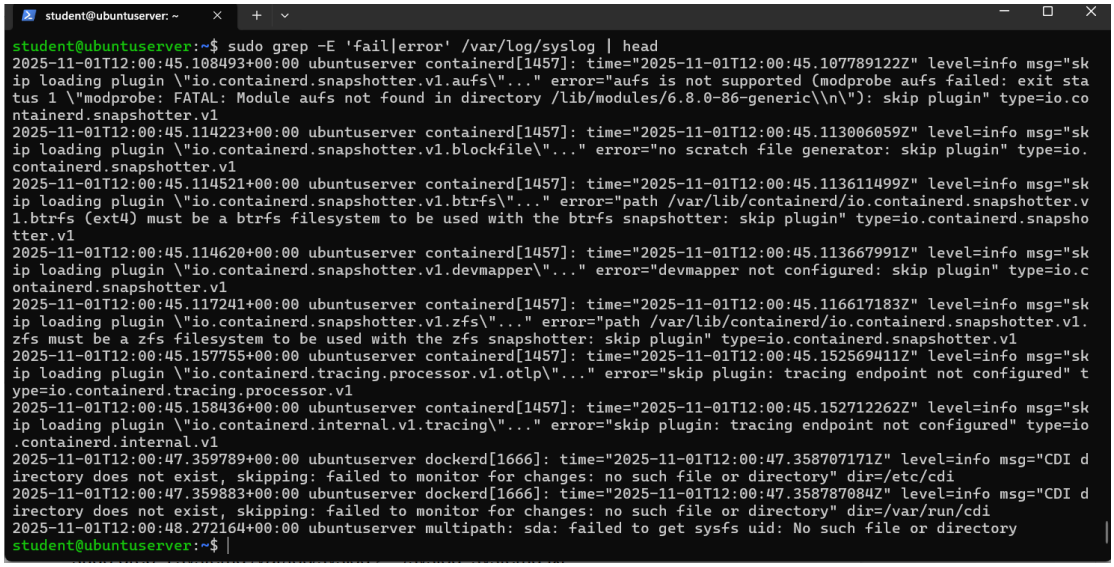
task9_grep_more.png



3. grep failures/errors:

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

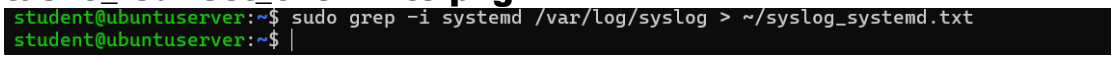
task9_grep_head.png



4. redirect:

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

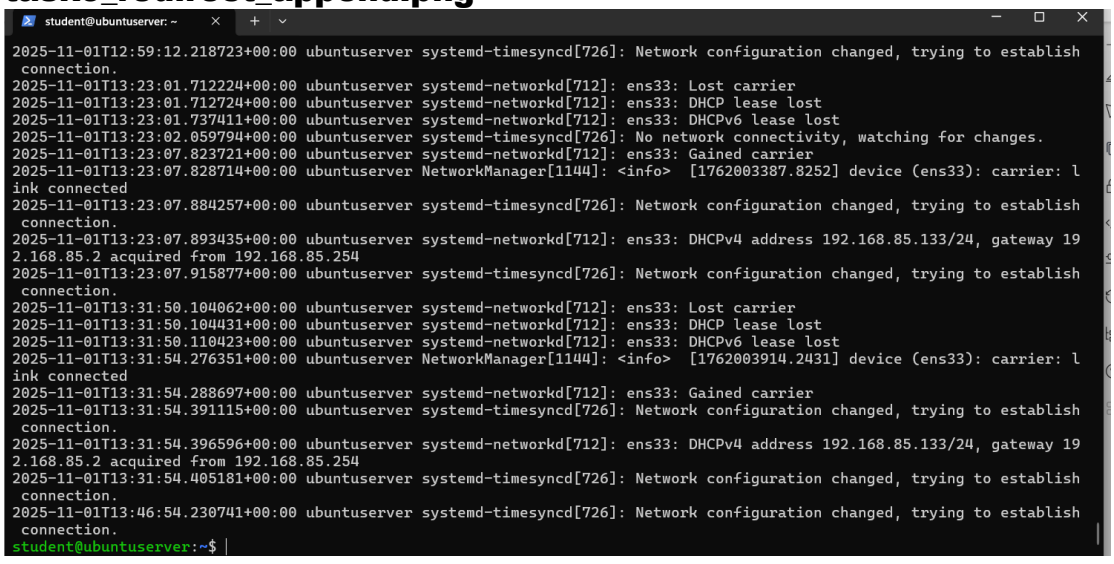
task9_redirect_overwrite.png



append:

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

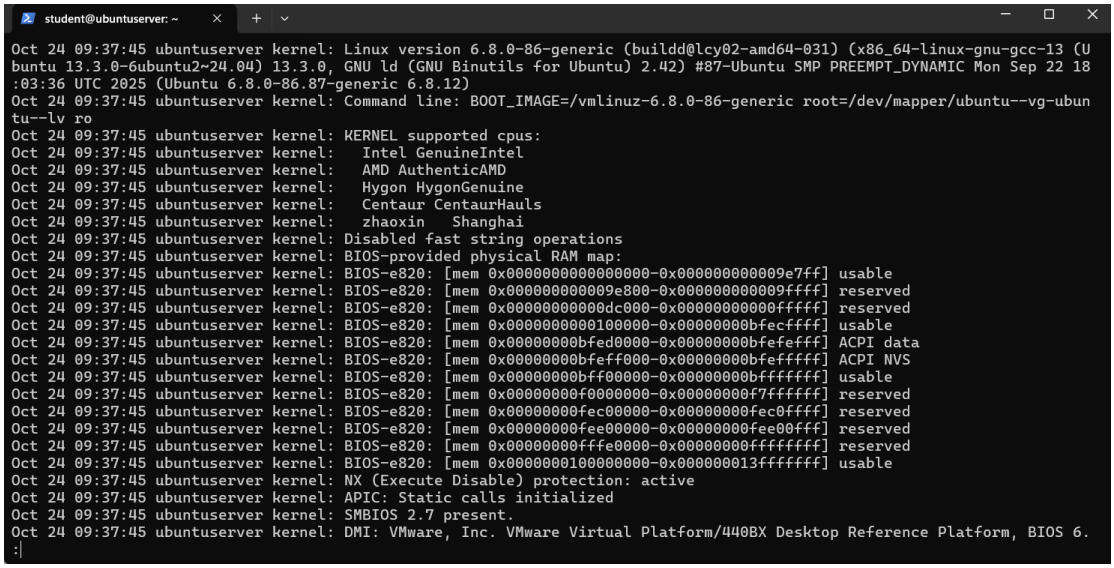
task9_redirect_append.png



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)

1. Include bash shebang

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

#!/bin/bash

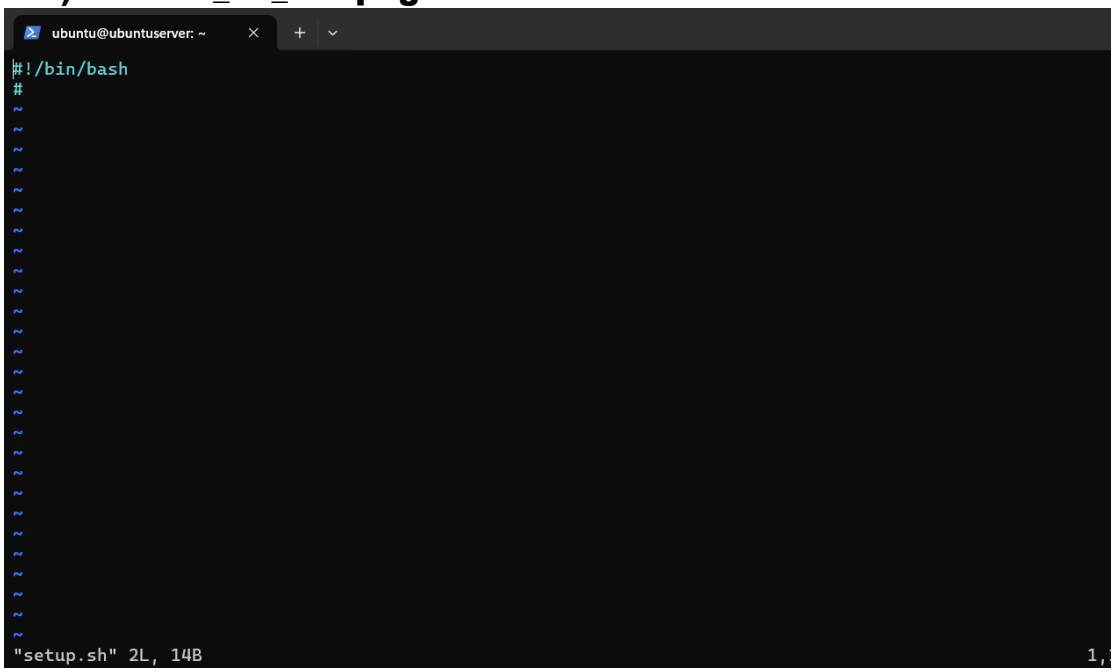
Steps:

vim setup.sh → add the shebang line → save and quit
chmod +x setup.sh
./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



```
ubuntu@ubuntuserver:~$ vim setup.sh
ubuntu@ubuntuserver:~$
```

```
ubuntu@ubuntuserver:~$ chmod +x setup.sh
./setup.sh
ubuntu@ubuntuserver:~$
```

2. Define variable var1 and echo it

Code to append:

```
# Define and show var1
var1="Hello from Lab 6"echo "var1: $var1"
```

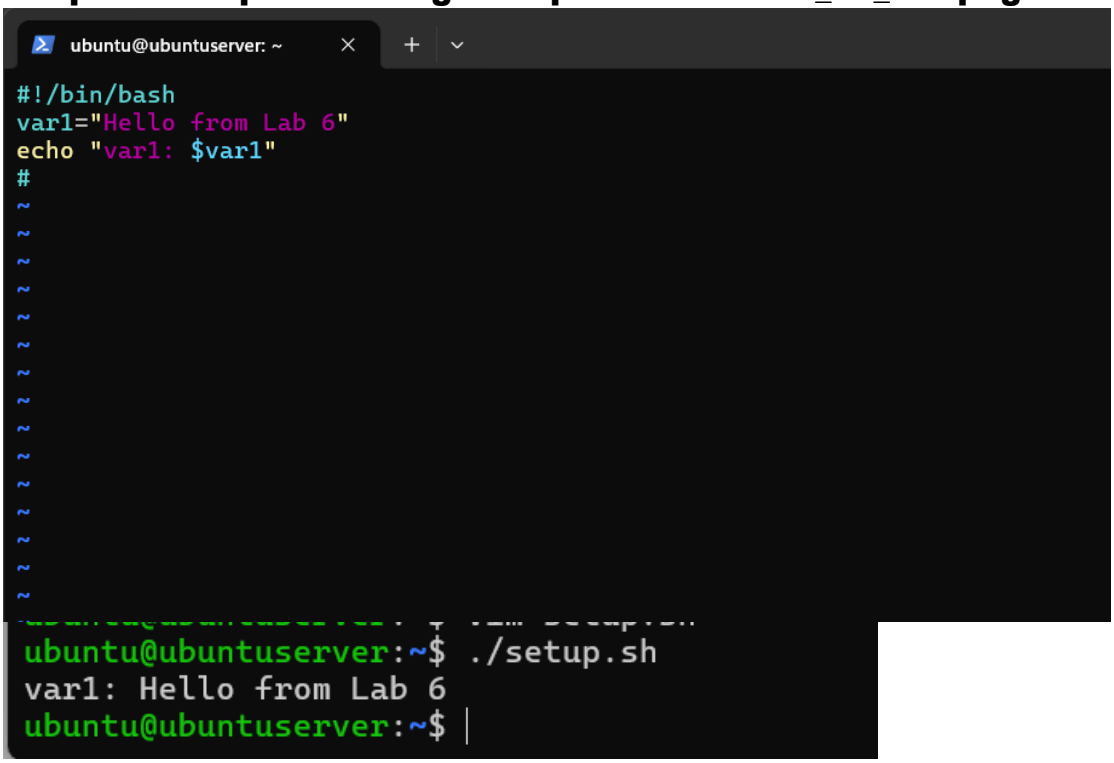
Steps:

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

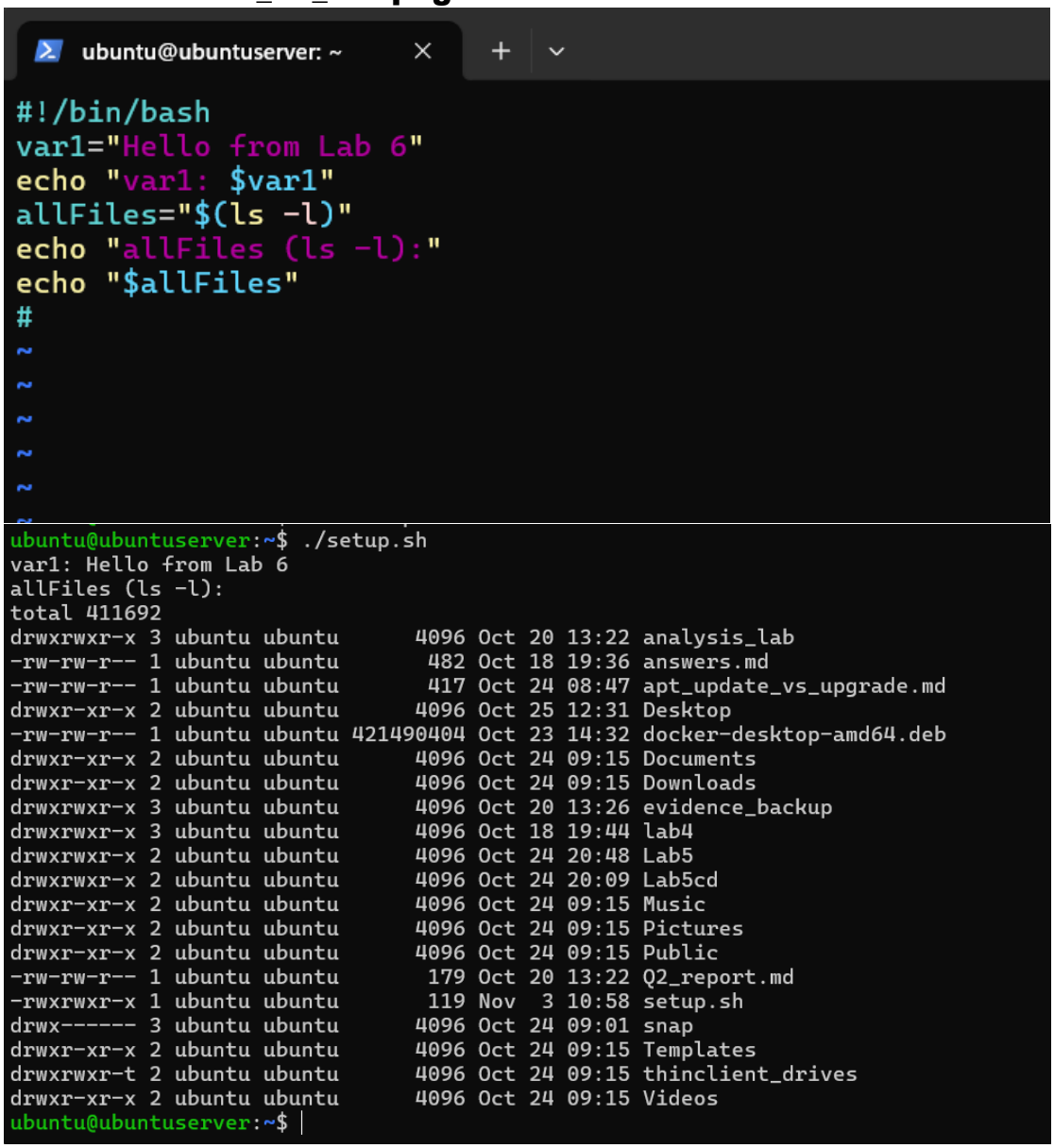
vim setup.sh → append the code above → save and quit
./setup.sh

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

Steps:

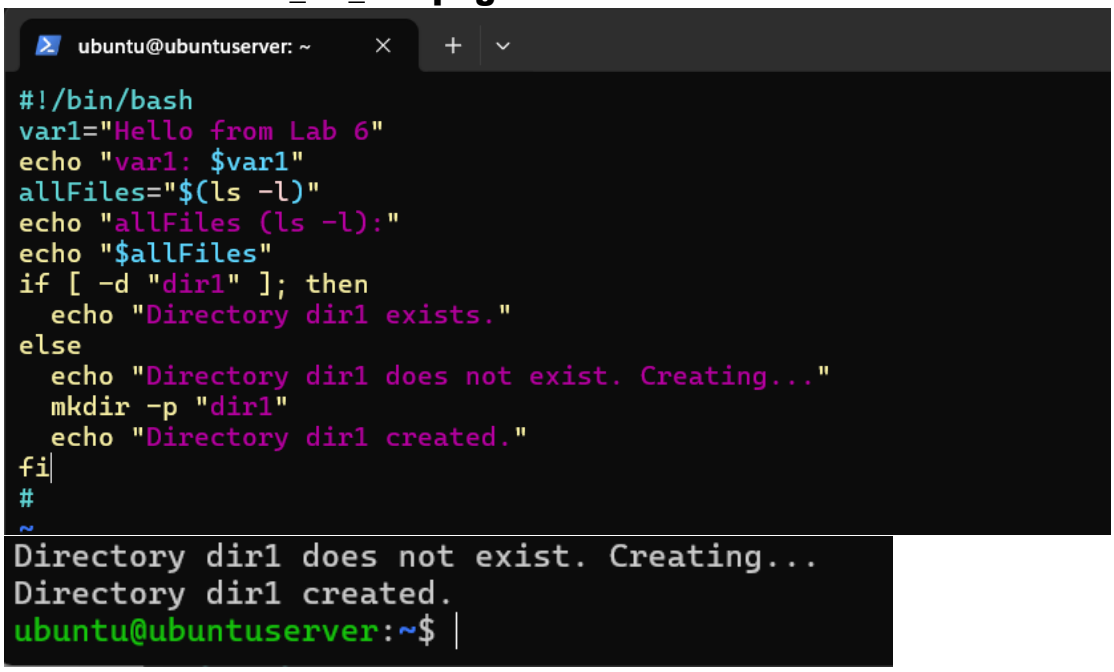
vim setup.sh → append the code above → save and quit

./setup.sh

Screenshots:

vim editor showing dir1 check code: task10_b4_vim.png

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 checkif [ -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
```

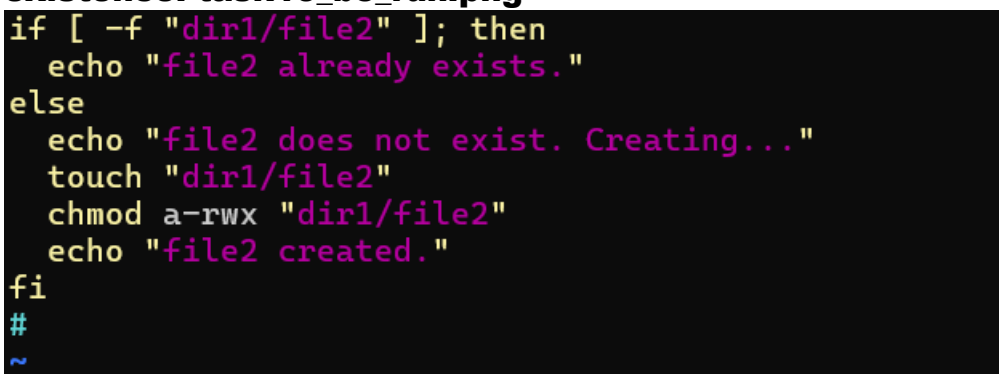
Steps:

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

Screenshots:

vim editor showing file2 check code: task10_b5_vim.png

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



```
ubuntu@ubuntuserver:~$ ./setup.sh
file2 does not exist. Creating...
file2 created.
ubuntu@ubuntuserver:~$
```

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:

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

Screenshots:

vim editor showing permission-check code: task10_b6_vim.png
script run output showing the permission grants and final ls -l dir1/file2: task10_b6_run.png

```
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"
#
-- INSERT --
```

```
ubuntu@ubuntuserver:~$ ./setup.sh
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 ubuntu ubuntu 0 Nov  3 11:04 dir1/file2
ubuntu@ubuntuserver:~$
```

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

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)

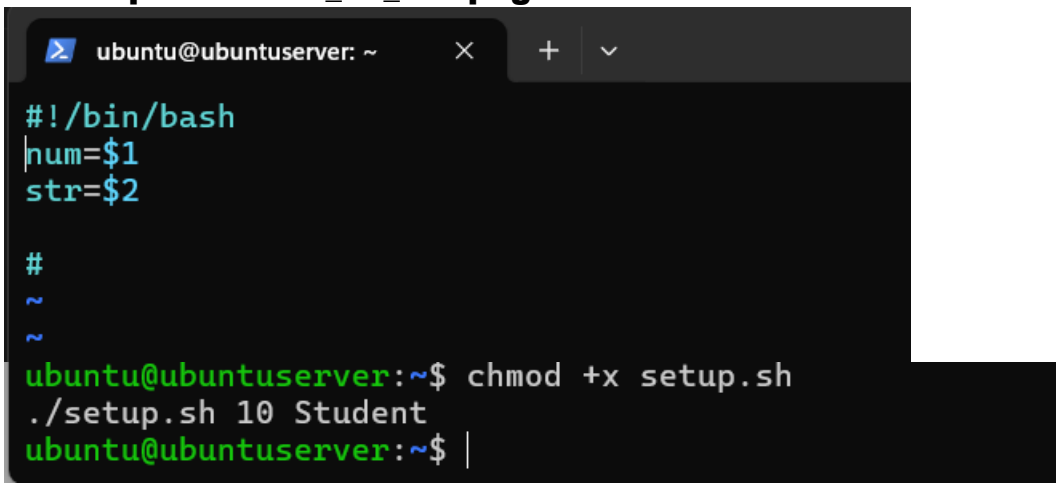
Make executable and run with examples:

```
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
```

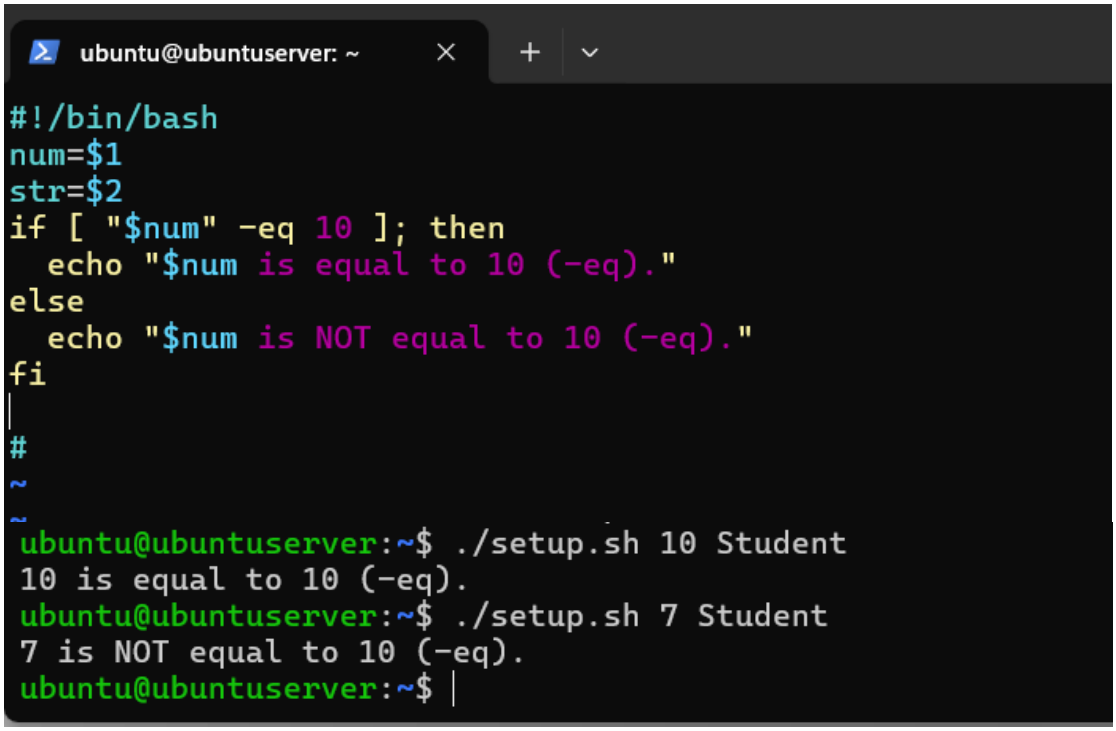
Save and quit; then run these commands (capture both in one terminal screenshot):

```
./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

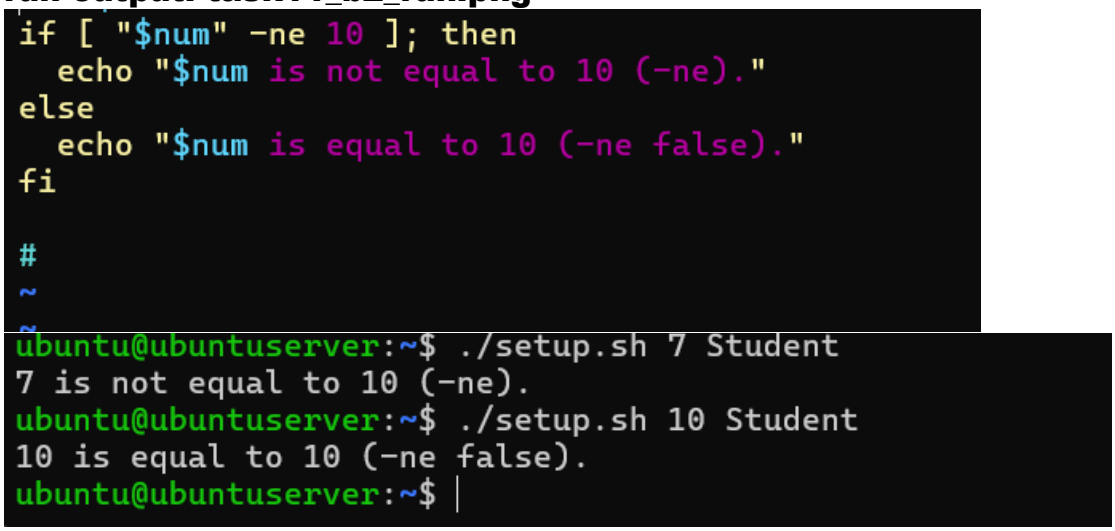
Save and quit; run:

./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

```
if [ "$num" -gt 10 ]; then
  echo "$num is greater than 10 (-gt)."
else
  echo "$num is NOT greater than 10 (-gt)."
fi
#
~
ubuntu@ubuntuserver:~$ ./setup.sh 12 Student
./setup.sh 9 Student
12 is greater than 10 (-gt).
9 is NOT greater than 10 (-gt).
ubuntu@ubuntuserver:~$
```

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

```
if [ "$num" -lt 10 ]; then
  echo "$num is less than 10 (-lt)."
else
  echo "$num is NOT less than 10 (-lt)."
fi
#
ubuntu@ubuntuserver:~$ ./setup.sh 5 Student
./setup.sh 11 Student
5 is less than 10 (-lt).
11 is NOT less than 10 (-lt).
ubuntu@ubuntuserver:~$
```

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

run output: task11_b5_run.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
```

```
ubuntu@ubuntuserver:~$ ./setup.sh 10 Student
./setup.sh 8 Student
10 is greater than or equal to 10 (-ge).
8 is NOT greater than or equal to 10 (-ge).
ubuntu@ubuntuserver:~$
```

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

run output: task11_b6_run.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
```

```
ubuntu@ubuntuserver:~$ ./setup.sh 10 Student
./setup.sh 12 Student
10 is less than or equal to 10 (-le).
12 is NOT less than or equal to 10 (-le).
ubuntu@ubuntuserver:~$
```

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

run output: task11_b7_run.png

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

```
ubuntu@ubuntuserver:~$ ./setup.sh 10 Student
./setup.sh 10 Test
Second argument equals 'Student' ( = ).
Second argument does NOT equal 'Student' ( = ).
ubuntu@ubuntuserver:~$
```

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

run output: task11_b8_run.png

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

#
~
~
```

```
ubuntu@ubuntuserver:~$ ./setup.sh 10 Test
./setup.sh 10 Student
Second argument is not equal to 'Student' ( != ).
Second argument equals 'Student' ( != false).
ubuntu@ubuntuserver:~$
```

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

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

```
ubuntu@ubuntuserver:~$ ./setup.sh 10
./setup.sh 10 Student
Second argument is empty (zero-length).
Second argument is not empty.
ubuntu@ubuntuserver:~$
```

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

```
ubuntu@ubuntuserver: ~
#!/bin/bash
# Script to demonstrate printing all user-entered arguments using $*
#
~
~
ubuntu@ubuntuserver:~$ chmod +x setup.sh
./setup.sh
ubuntu@ubuntuserver:~$
```

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
script run output showing the printed arguments: task12_b2_run.png

```
# Print all arguments using $*
echo "Printing all arguments using \$*:"
for arg in $*; do
  echo "Argument: $arg"
done
#
```

```
ubuntu@ubuntuserver:~$ ./setup.sh one "two words" three
Printing all arguments using $*:
Argument: one
Argument: two
Argument: words
Argument: three
ubuntu@ubuntuserver:~$
```

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

```
ubuntu@ubuntuserver: ~
#!/bin/bash

#
~
~
~
```

```
ubuntu@ubuntuserver:~$ chmod +x setup.sh
./setup.sh
ubuntu@ubuntuserver:~$
```

2. Add the while-loop summation (interactive)

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

```
# While-loop summation (interactive)
sum=0while 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"doneecho "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

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

```
# 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"


#
```

```
ubuntu@ubuntuserver:~$ ./setup.sh
Enter a number (or 'q' to quit): 5
Total Score: 5
Enter a number (or 'q' to quit): 3
Total Score: 8
Enter a number (or 'q' to quit): q
Final total: 8
ubuntu@ubuntuserver:~$
```

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 interactivelysum_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 functionecho "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.

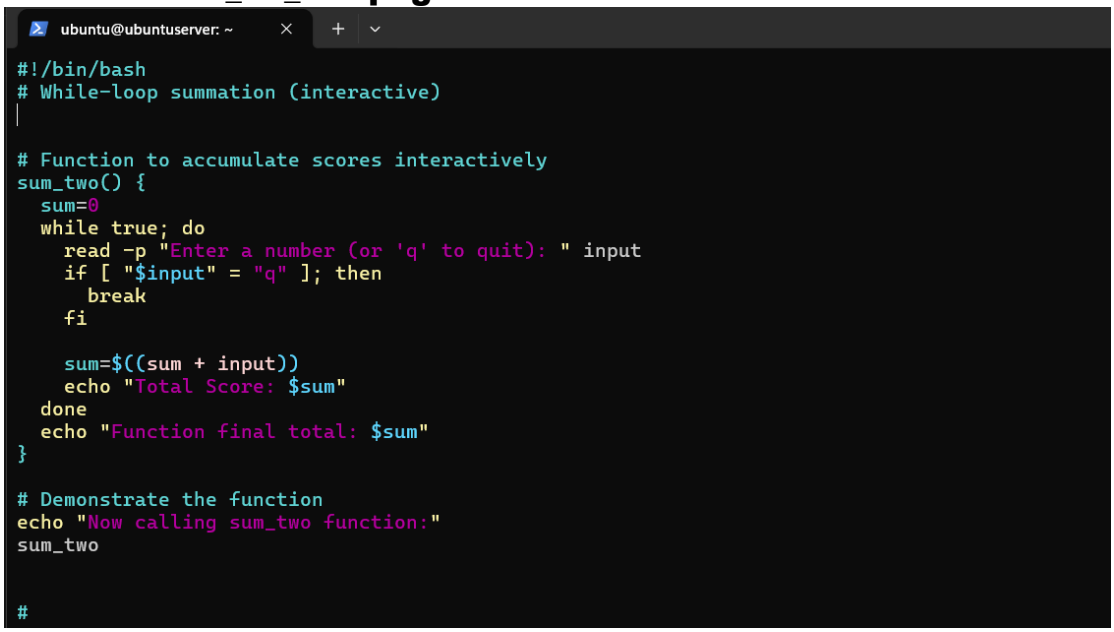
Run the script and demonstrate a short session (example): enter 3, 4, q when prompted by the function:

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

Screenshots:

vim editor showing function appended: task13_b3_vim.png

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



```
ubuntu@ubuntuserver:~$ ./setup.sh
Now calling sum_two function:
Enter a number (or 'q' to quit): 4
Total Score: 4
Enter a number (or 'q' to quit): 2
Total Score: 6
Enter a number (or 'q' to quit): q
Function final total: 6
ubuntu@ubuntuserver:~$
```

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

resultsum_args() {
  a=$1
  b=$2
  return $((a + b))
}

 # Demonstrate sum_args functionecho "Now demonstrating sum_args function:"

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

Save and quit (:wq)

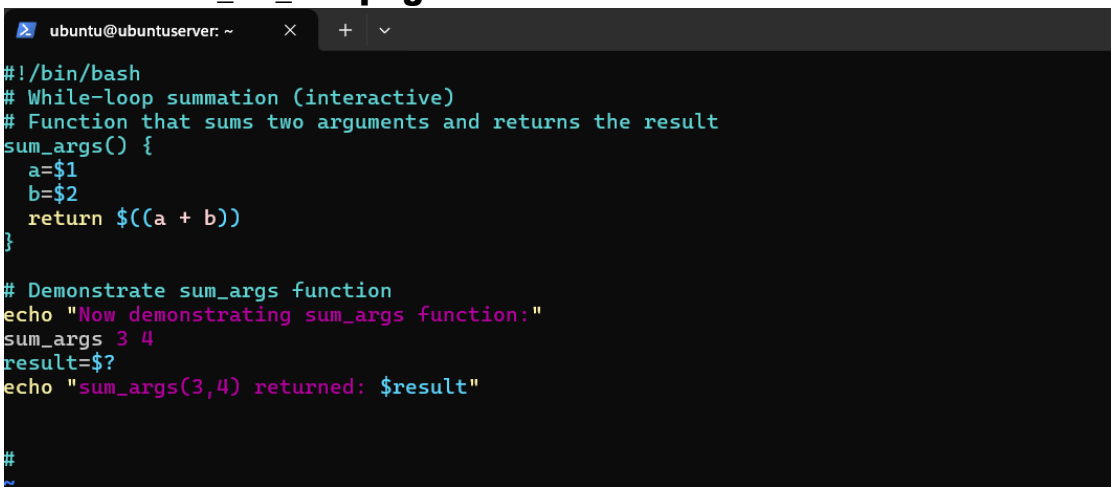
Run the script and capture the demonstration output:

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

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



```
ubuntu@ubuntuserver:~$ ./setup.sh
Now demonstrating sum_args function:
sum_args(3,4) returned: 7
```

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

Steps:

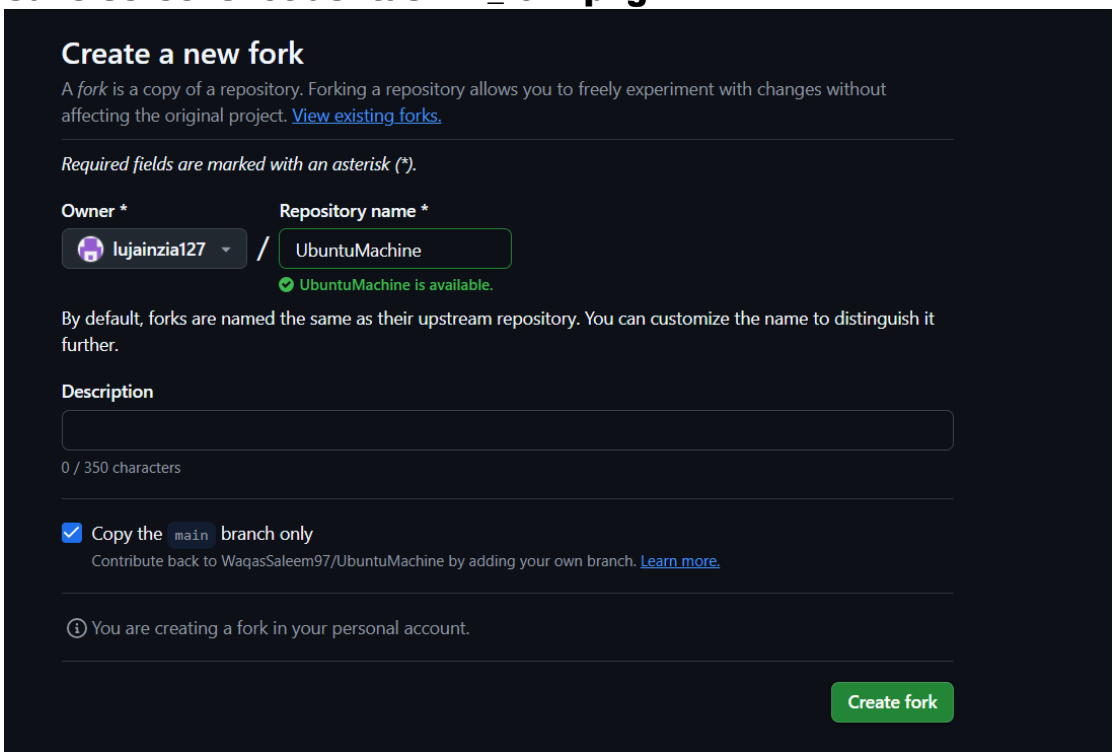
1. Fork the repository to your GitHub account

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

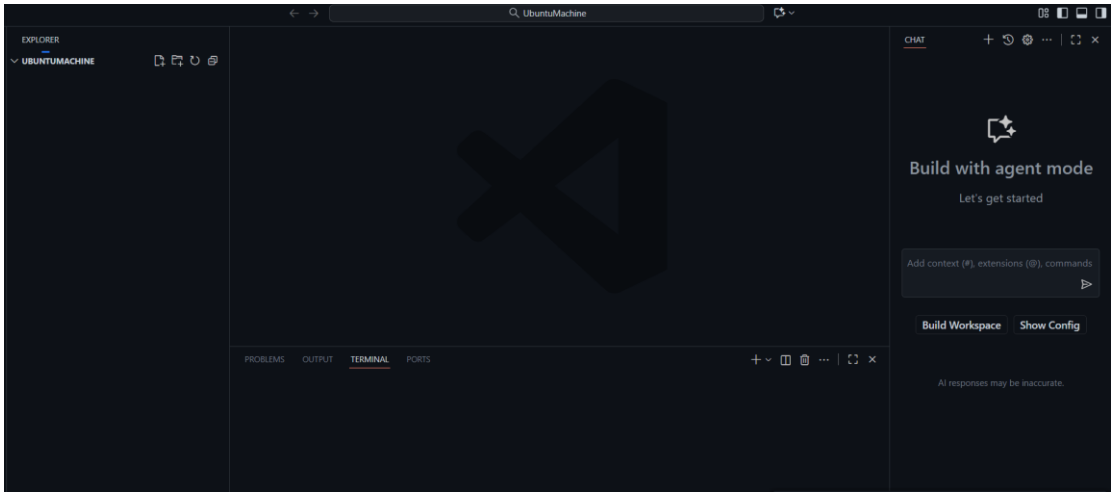


2. Open a Codespace on your fork

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

```
@lujainzia127 ➜/workspaces/UbuntuMachine (main) $ ls -l start-desktop.sh stop-desktop.sh
-rwxrwxrwx 1 codespace root 1333 Nov  3 12:47 start-desktop.sh
-rwxrwxrwx 1 codespace root  428 Nov  3 12:47 stop-desktop.sh
@lujainzia127 ➜/workspaces/UbuntuMachine (main) $
```

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

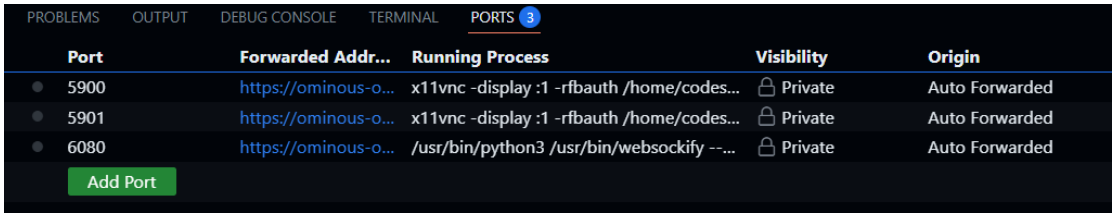
```
@lujainzia127 ➜/workspaces/UbuntuMachine (main) $ WebSocket server settings:
  - Listen on :6080
  - Web server. Web root: /usr/share/novnc
  - No SSL/TLS support (no cert file)
  - proxying from :6080 to localhost:5901
```

5. Verify forwarded ports in Codespaces (Ports view)

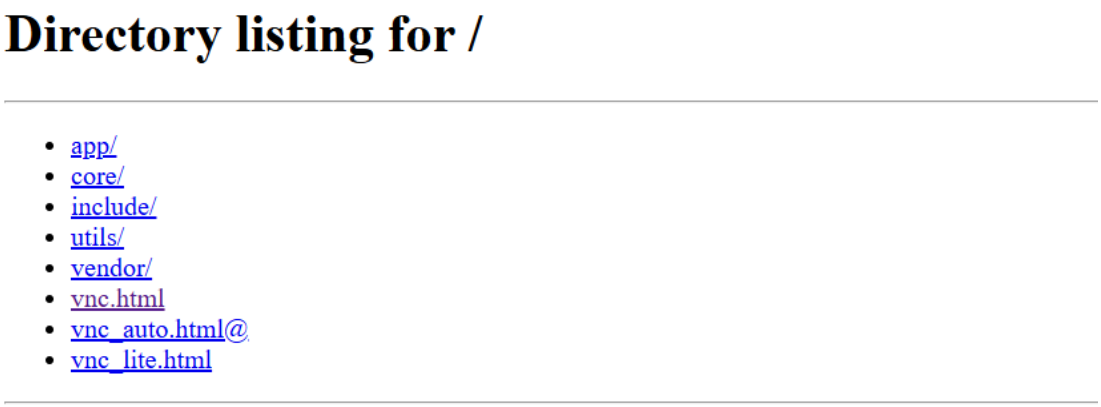
Open the Codespaces "Ports" panel / view and confirm port 6080 is forwarded and visible.

Save a screenshot of the Ports view showing port 6080 and its status:

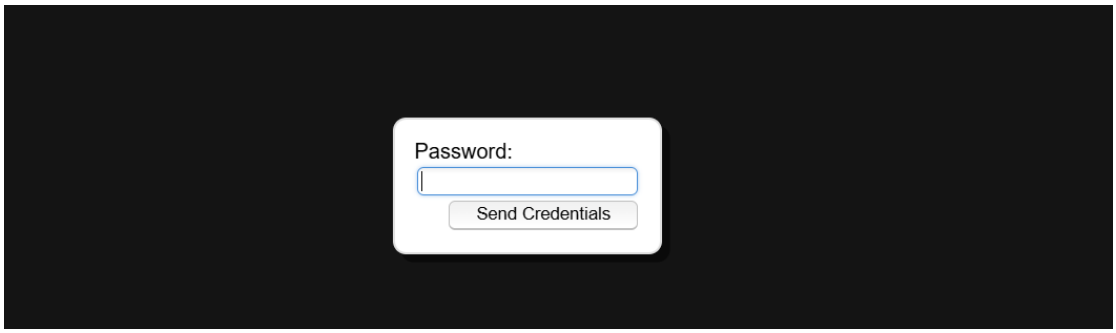
task14_ports_view.png



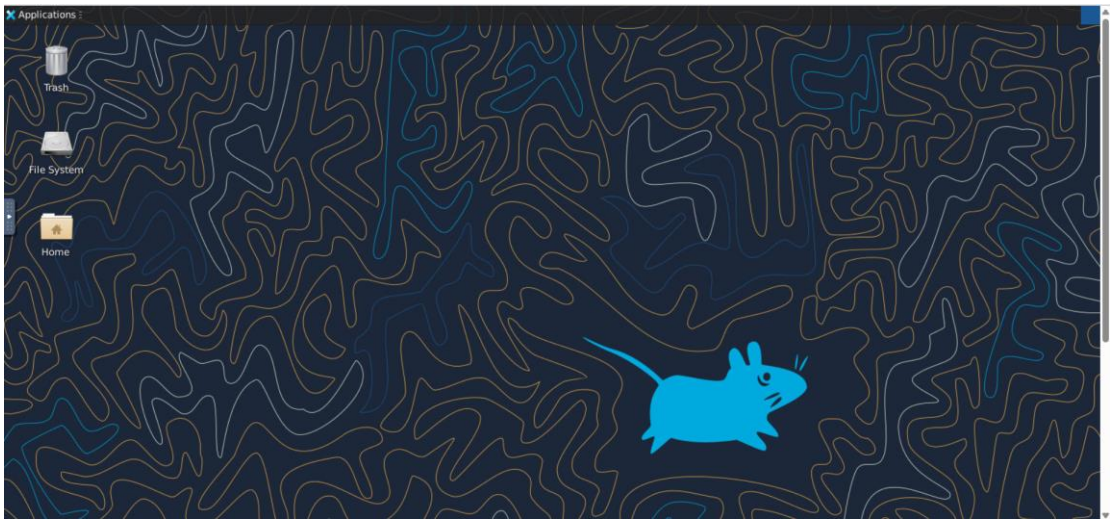
6. Open forwarded port 6080 and connect to VNC HTML page
In the Codespaces UI, open the forwarded port's preview URL or copy the forwarded URL and open it in your browser.
Visit the port 6080 address and click the vnc.html link.
When prompted for a password enter:
codespace
Capture screenshots of:
The 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



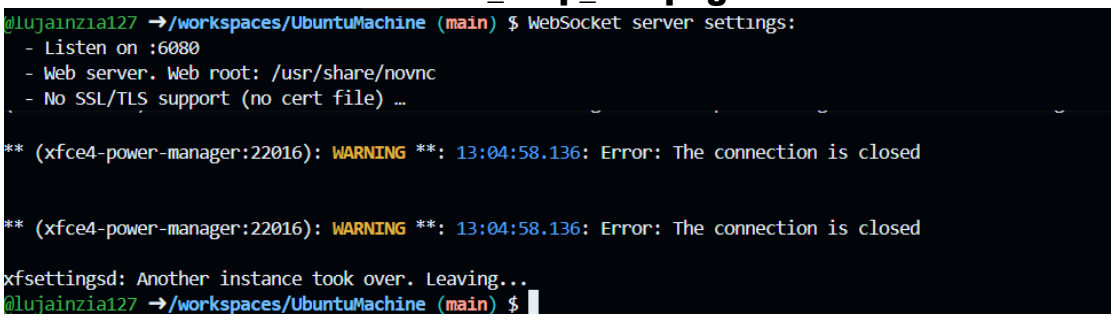
(Optional) A focused screenshot of vnc.html UI showing the "Connect" button before/after connecting: task14_vnc_connect.png

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 question:

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:
Create groups g1, g2, and g3.
Screenshot: Q1_groups_created.png

```
ubuntu@ubuntuserver:~$ sudo groupadd g1
sudo groupadd g2
sudo groupadd g3
[sudo] password for ubuntu:
ubuntu@ubuntuserver:~$
```

Change examuser's primary group to g3 and add g1 and g2 as supplementary groups.
Screenshot: Q1_group_changes.png

```
ubuntu@ubuntuserver:~$ sudo usermod -g g3 examuser
ubuntu@ubuntuserver:~$ sudo usermod -aG g1,g2 examuser
ubuntu@ubuntuserver:~$
```

Show the final id and /etc/group lines that prove the changes.
Screenshot: Q1_group_verification.png

```
ubuntu@ubuntuserver:~$ id examuser
grep 'g[123]' /etc/group
uid=1009(examuser) gid=1008(g3) groups=1008(g3),100(users),1006(g1),1007(g2)
g1:x:1006:examuser
g2:x:1007:examuser
g3:x:1008:
ubuntu@ubuntuserver:~$
```

---

2. Ownership and Permission Tasks

Scenario:
Demonstrate ownership changes and apply both symbolic and numeric permission changes.
Steps:
Create workspace/secret.txt, change its owner to examuser and group to g1.
Screenshot: Q2_chown_chgrp.png

```
ubuntu@ubuntuserver:~$ ls -l ~/workspace/secret.txt
-rw-rw-r-- 1 examuser g1 0 Nov  3 13:12 /home/ubuntu/workspace/secret.txt
ubuntu@ubuntuserver:~$
```

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

```
ubuntu@ubuntuserver:~$ sudo chmod 700 ~/workspace/secret.txt
ubuntu@ubuntuserver:~$ sudo chmod go-rwx ~/workspace/secret.txt
ubuntu@ubuntuserver:~$ ls -l ~/workspace/secret.txt
-rwx------ 1 examuser g1 0 Nov  3 13:12 /home/ubuntu/workspace/secret.txt
ubuntu@ubuntuserver:~$
```

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

Screenshot: Q2_permissions_ls.png

```
ubuntu@ubuntuserver:~$ ls -l ~/workspace/secret.txt
-rwx------ 1 examuser g1 0 Nov  3 13:12 /home/ubuntu/workspace/secret.txt
ubuntu@ubuntuserver:~$
```

---

3. Pipes, Grep, and Redirection Practice

Scenario:

Filter system logs and save results using redirection and piping.

Steps:

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

```
ubuntu@ubuntuserver:~$ grep -Ei 'error|fail' /var/log/syslog | head -n 20
2025-11-01T12:00:45.108493+00:00 ubuntuserver containerd[1457]: time="2025-11-01T12:00:45.107789122Z" level=info msg="sk
ip loading plugin \"io.containerd.snapshotter.v1.aufs\"..." error="aufs is not supported (modprobe aufs failed: exit sta
tus 1 \"modprobe: FATAL: Module aufs not found in directory /lib/modules/6.8.0-86-generic\\n\"): skip plugin" type=io.co
ntainerd.snapshotter.v1
2025-11-01T12:00:45.114223+00:00 ubuntuserver containerd[1457]: time="2025-11-01T12:00:45.113006059Z" level=info msg="sk
ip loading plugin \"io.containerd.snapshotter.v1.blockfile\"..." error="no scratch file generator: skip plugin" type=io.
containerd.snapshotter.v1
2025-11-01T12:00:45.114521+00:00 ubuntuserver containerd[1457]: time="2025-11-01T12:00:45.113611499Z" level=info msg="sk
ip loading plugin \"io.containerd.snapshotter.v1.btrfs\"..." error="path /var/lib/containerd/io.containerd.snapshotter.v
1.btrfs (ext4) must be a btrfs filesystem to be used with the btrfs snapshotter: skip plugin" type=io.containerd.snapsho
tter.v1
2025-11-01T12:00:45.114620+00:00 ubuntuserver containerd[1457]: time="2025-11-01T12:00:45.113667991Z" level=info msg="sk
ip loading plugin \"io.containerd.snapshotter.v1.devmapper\"..." error="devmapper not configured: skip plugin" type=io.c
ontainerd.snapshotter.v1
2025-11-01T12:00:45.117241+00:00 ubuntuserver containerd[1457]: time="2025-11-01T12:00:45.116617183Z" level=info msg="sk
ip loading plugin \"io.containerd.snapshotter.v1.zfs\"..." error="path /var/lib/containerd/io.containerd.snapshotter.v1.
zfs must be a zfs filesystem to be used with the zfs snapshotter: skip plugin" type=io.containerd.snapshotter.v1
2025-11-01T12:00:45.157755+00:00 ubuntuserver containerd[1457]: time="2025-11-01T12:00:45.152569411Z" level=info msg="sk
ip loading plugin \"io.containerd.tracing.processor.v1.otlp\"..." error="skip plugin: tracing endpoint not configured" t
ype=io.containerd.tracing.processor.v1
2025-11-01T12:00:45.158436+00:00 ubuntuserver containerd[1457]: time="2025-11-01T12:00:45.152712262Z" level=info msg="sk
ip loading plugin \"io.containerd.internal.v1.tracing\"..." error="skip plugin: tracing endpoint not configured" type=io
.containerd.internal.v1
2025-11-01T12:00:47.359789+00:00 ubuntuserver dockerd[1666]: time="2025-11-01T12:00:47.358707171Z" level=info msg="CDI d
irectory does not exist, skipping: failed to monitor for changes: no such file or directory" dir=/etc/cdi
2025-11-01T12:00:47.359883+00:00 ubuntuserver dockerd[1666]: time="2025-11-01T12:00:47.358787084Z" level=info msg="CDI d
irectory does not exist, skipping: failed to monitor for changes: no such file or directory" dir=/var/run/cdi
2025-11-01T12:00:48.272164+00:00 ubuntuserver multipath: sda: failed to get sysfs uid: No such file or directory
```

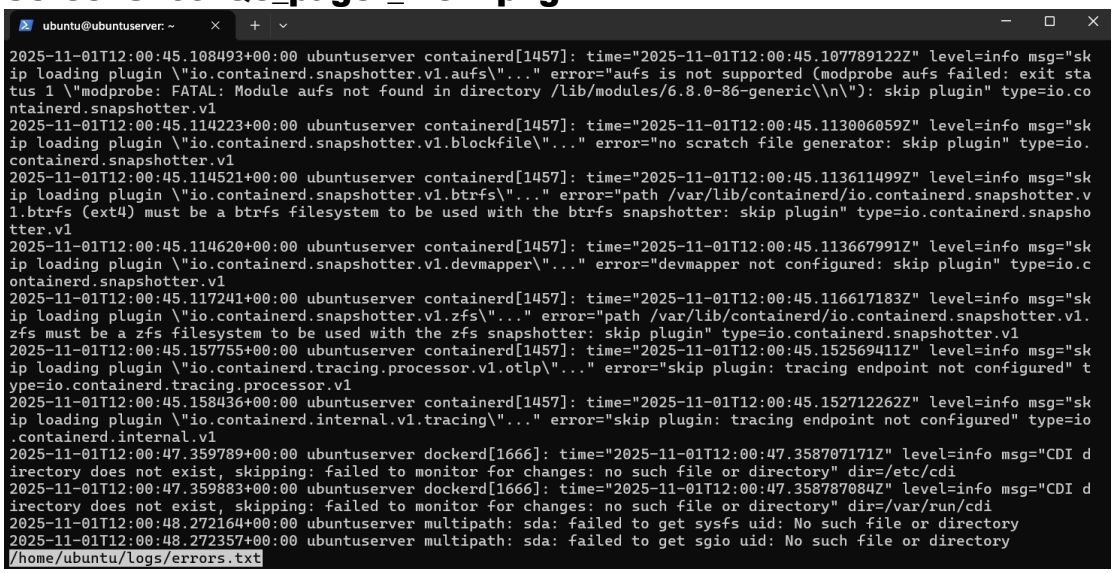
Save the filtered results to a file ~/logs/errors.txt using overwrite, then append additional matching lines using append redirection.

Screenshot: Q3_redirect_overwrite_append.png

```
ubuntu@ubuntuserver:~$ mkdir -p ~/logs
grep -Ei 'error|fail' /var/log/syslog > ~/logs/errors.txt
ubuntu@ubuntuserver:~$ grep -Ei 'warn|critical' /var/log/syslog >> ~/logs/errors.txt
ubuntu@ubuntuserver:~$
```

Use a pager to view the saved file.

Screenshot: Q3_pager_view.png



```
ubuntu@ubuntuserver: ~
2025-11-01T12:00:45.108493+00:00 ubuntuserver containerd[1457]: time="2025-11-01T12:00:45.107789122Z" level=info msg="sk
ip loading plugin \"io.containerd.snapshotter.v1.aufs\"..." error="aufs is not supported (modprobe aufs failed: exit sta
tus 1 \"modprobe: FATAL: Module aufs not found in directory /lib/modules/6.8.0-86-generic\\n\"): skip plugin" type=io.co
ntainerd.snapshotter.v1
2025-11-01T12:00:45.114223+00:00 ubuntuserver containerd[1457]: time="2025-11-01T12:00:45.113006059Z" level=info msg="sk
ip loading plugin \"io.containerd.snapshotter.v1.blockfile\"..." error="no scratch file generator: skip plugin" type=io.
containerd.snapshotter.v1
2025-11-01T12:00:45.114521+00:00 ubuntuserver containerd[1457]: time="2025-11-01T12:00:45.113611499Z" level=info msg="sk
ip loading plugin \"io.containerd.snapshotter.v1.btrfs\"..." error="path /var/lib/containerd/io.containerd.snapshotter.v
1.btrfs (ext4) must be a btrfs filesystem to be used with the btrfs snapshotter: skip plugin" type=io.containerd.snapsho
tter.v1
2025-11-01T12:00:45.114620+00:00 ubuntuserver containerd[1457]: time="2025-11-01T12:00:45.113667991Z" level=info msg="sk
ip loading plugin \"io.containerd.snapshotter.v1.devmapper\"..." error="devmapper not configured: skip plugin" type=io.c
ontainerd.snapshotter.v1
2025-11-01T12:00:45.117241+00:00 ubuntuserver containerd[1457]: time="2025-11-01T12:00:45.116617183Z" level=info msg="sk
ip loading plugin \"io.containerd.snapshotter.v1.zfs\"..." error="path /var/lib/containerd/io.containerd.snapshotter.v1.
zfs must be a zfs filesystem to be used with the zfs snapshotter: skip plugin" type=io.containerd.snapshotter.v1
2025-11-01T12:00:45.157755+00:00 ubuntuserver containerd[1457]: time="2025-11-01T12:00:45.152569411Z" level=info msg="sk
ip loading plugin \"io.containerd.tracing.processor.v1.otlp\"..." error="skip plugin: tracing endpoint not configured" t
ype=io.containerd.tracing.processor.v1
2025-11-01T12:00:45.158436+00:00 ubuntuserver containerd[1457]: time="2025-11-01T12:00:45.152712262Z" level=info msg="sk
ip loading plugin \"io.containerd.internal.v1.tracing\"..." error="skip plugin: tracing endpoint not configured" type=io
.containerd.internal.v1
2025-11-01T12:00:47.359789+00:00 ubuntuserver dockerd[1666]: time="2025-11-01T12:00:47.358707171Z" level=info msg="CDI d
irectory does not exist, skipping: failed to monitor for changes: no such file or directory" dir=/etc/cdi
2025-11-01T12:00:47.359883+00:00 ubuntuserver dockerd[1666]: time="2025-11-01T12:00:47.358787084Z" level=info msg="CDI d
irectory does not exist, skipping: failed to monitor for changes: no such file or directory" dir=/var/run/cdi
2025-11-01T12:00:48.272164+00:00 ubuntuserver multipath: sda: failed to get sysfs uid: No such file or directory
2025-11-01T12:00:48.272357+00:00 ubuntuserver multipath: sda: failed to get sgio uid: No such file or directory
/home/ubuntu/logs/errors.txt
```

---

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

Scenario:

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

Steps:

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

Screenshot: Q4_step1_var1.png

```
ubuntu@ubuntuserver:~$ vim setup.sh
ubuntu@ubuntuserver:~$ chmod +x setup.sh
./setup.sh
Hello from var1
ubuntu@ubuntuserver:~$
```

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

Screenshot: Q4_step2_allfiles.png

```
ubuntu@ubuntuserver:~$ ./setup.sh
All files listed:
total 411704
drwxrwxr-x 3 ubuntu ubuntu      4096 Oct 20 13:22 analysis_lab
-rw-rw-r-- 1 ubuntu ubuntu       482 Oct 18 19:36 answers.md
-rw-rw-r-- 1 ubuntu ubuntu       417 Oct 24 08:47 apt_update_vs_upgrade.md
drwxr-xr-x 2 ubuntu ubuntu      4096 Oct 25 12:31 Desktop
drwxrwxr-x 2 ubuntu ubuntu      4096 Nov  3 11:04 dir1
-rw-rw-r-- 1 ubuntu ubuntu 421490404 Oct 23 14:32 docker-desktop-amd64.deb
drwxr-xr-x 2 ubuntu ubuntu      4096 Oct 24 09:15 Documents
drwxr-xr-x 2 ubuntu ubuntu      4096 Oct 24 09:15 Downloads
drwxrwxr-x 3 ubuntu ubuntu      4096 Oct 20 13:26 evidence_backup
drwxrwxr-x 3 ubuntu ubuntu      4096 Oct 18 19:44 lab4
drwxrwxr-x 2 ubuntu ubuntu      4096 Oct 24 20:48 Lab5
drwxrwxr-x 2 ubuntu ubuntu      4096 Oct 24 20:09 Lab5cd
drwxrwxr-x 2 ubuntu ubuntu      4096 Nov  3 13:27 logs
drwxr-xr-x 2 ubuntu ubuntu      4096 Oct 24 09:15 Music
drwxr-xr-x 2 ubuntu ubuntu      4096 Oct 24 09:15 Pictures
drwxr-xr-x 2 ubuntu ubuntu      4096 Oct 24 09:15 Public
-rw-rw-r-- 1 ubuntu ubuntu       179 Oct 20 13:22 Q2_report.md
-rwxrwxr-x 1 ubuntu ubuntu        73 Nov  3 13:40 setup.sh
drwx------ 3 ubuntu ubuntu      4096 Oct 24 09:01 snap
drwxr-xr-x 2 ubuntu ubuntu      4096 Oct 24 09:15 Templates
drwxrwxr-t 2 ubuntu ubuntu      4096 Oct 24 09:15 thinclient_drives
drwxr-xr-x 2 ubuntu ubuntu      4096 Oct 24 09:15 Videos
drwxrwxr-x 2 ubuntu ubuntu      4096 Nov  3 13:12 workspace
ubuntu@ubuntuserver:~$
```

Append directory and file checks that
create dir1 and dir1/file2 if missing, and display their final permissions.
Screenshot: Q4_step3_dirfile_checks.png

```
All files listed:
total 411704
drwxrwxr-x 3 ubuntu ubuntu      4096 Oct 20 13:22 analysis_lab
-rw-rw-r-- 1 ubuntu ubuntu       482 Oct 18 19:36 answers.md
-rw-rw-r-- 1 ubuntu ubuntu       417 Oct 24 08:47 apt_update_vs_upgrade.md
drwxr-xr-x 2 ubuntu ubuntu      4096 Oct 25 12:31 Desktop
drwxrwxr-x 2 ubuntu ubuntu      4096 Nov  3 11:04 dir1
-rw-rw-r-- 1 ubuntu ubuntu 421490404 Oct 23 14:32 docker-desktop-amd64.deb
drwxr-xr-x 2 ubuntu ubuntu      4096 Oct 24 09:15 Documents
drwxr-xr-x 2 ubuntu ubuntu      4096 Oct 24 09:15 Downloads
drwxrwxr-x 3 ubuntu ubuntu      4096 Oct 20 13:26 evidence_backup
drwxrwxr-x 3 ubuntu ubuntu      4096 Oct 18 19:44 lab4
drwxrwxr-x 2 ubuntu ubuntu      4096 Oct 24 20:48 Lab5
drwxrwxr-x 2 ubuntu ubuntu      4096 Oct 24 20:09 Lab5cd
drwxrwxr-x 2 ubuntu ubuntu      4096 Nov  3 13:27 logs
drwxr-xr-x 2 ubuntu ubuntu      4096 Oct 24 09:15 Music
drwxr-xr-x 2 ubuntu ubuntu      4096 Oct 24 09:15 Pictures
drwxr-xr-x 2 ubuntu ubuntu      4096 Oct 24 09:15 Public
-rw-rw-r-- 1 ubuntu ubuntu       179 Oct 20 13:22 Q2_report.md
-rwxrwxr-x 1 ubuntu ubuntu       288 Nov  3 13:42 setup.sh
drwx------ 3 ubuntu ubuntu      4096 Oct 24 09:01 snap
drwxr-xr-x 2 ubuntu ubuntu      4096 Oct 24 09:15 Templates
drwxrwxr-t 2 ubuntu ubuntu      4096 Oct 24 09:15 thinclient_drives
drwxr-xr-x 2 ubuntu ubuntu      4096 Oct 24 09:15 Videos
drwxrwxr-x 2 ubuntu ubuntu      4096 Nov  3 13:12 workspace
Permissions of dir1:
drwxrwxr-x 2 ubuntu ubuntu 4096 Nov  3 11:04 dir1
Permissions of file2:
-rwx------ 1 ubuntu ubuntu 0 Nov  3 11:04 dir1/file2
ubuntu@ubuntuserver:~$
```

---

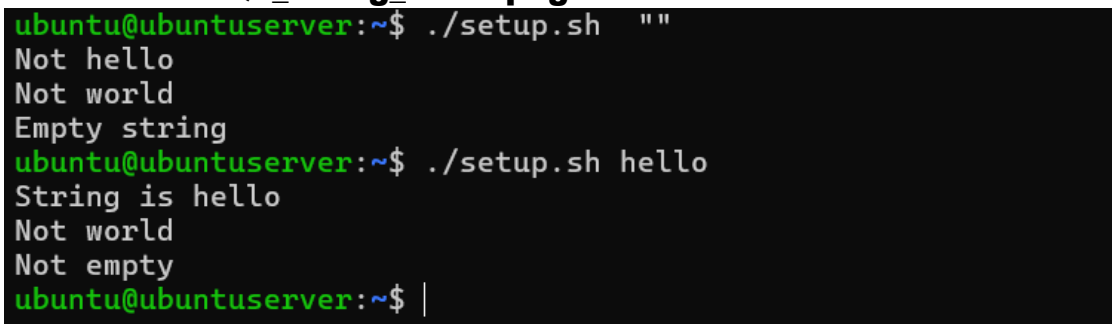
5. Script: Comparisons and String Tests

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

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

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

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

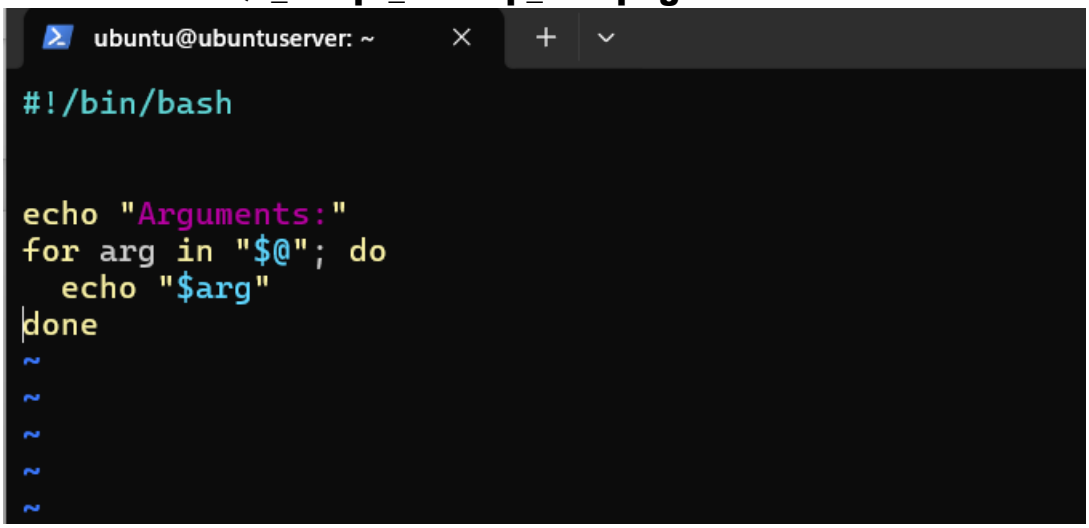


---

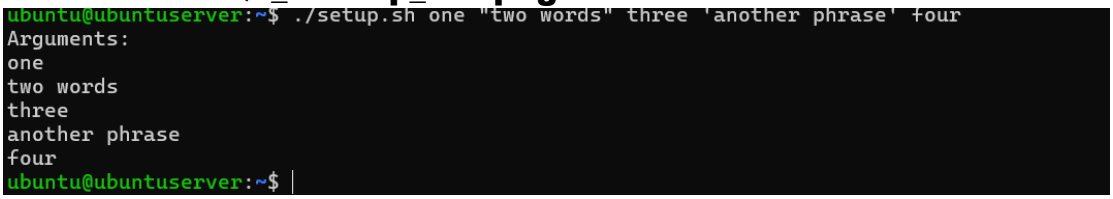
6. Script: For Loop and Argument Handling

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

Create/overwrite setup.sh to print every argument using "$@" in a for loop and save the file.
Screenshot: Q6_script_forloop_vim.png



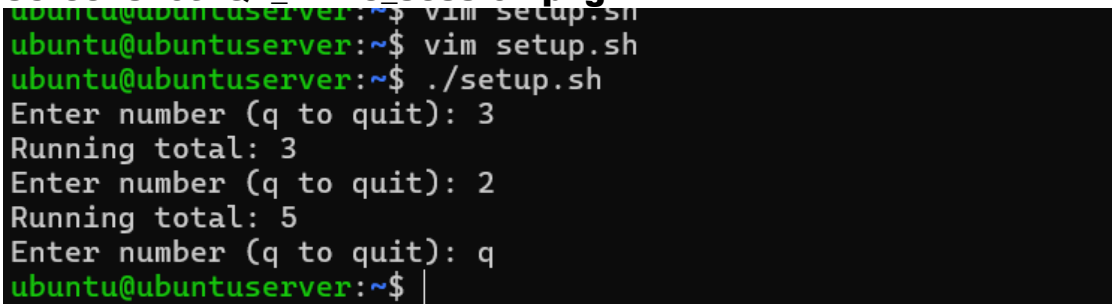
Run the script with mixed single and quoted multi-word arguments and capture the output showing each argument on its own line.
Screenshot: Q6_forloop_run.png



---

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:
Write an interactive while-loop that accumulates numbers until q is entered and shows running totals.
Screenshot: Q7_while_session.png



Add a function that accepts two numeric arguments, returns their sum, and demonstrate capturing its result in a variable. Screenshot: Q7_function_sum.png