Fatima Jinnah Women University

*Department of Software Engineering*



**SUBJECT: CC LAB**

**SUBMITTED TO: SIR SHOAIB**

**SUBMITTED BY: ZUNAIRA NOOR**

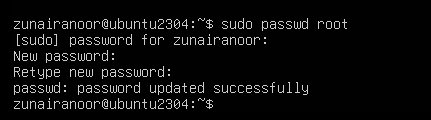
**REGISTRATION NO: 2023-BSE-075**

**SEMESTER: V-B**

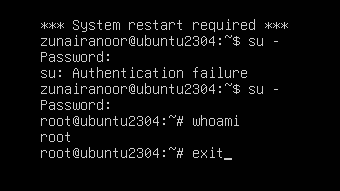
**Task 1 – Switch to root with su - and back to a normal user**

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

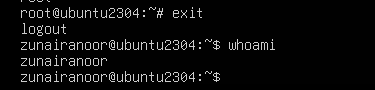
1. Set a root password (Ubuntu root is disabled by default; this enables su - temporarily for the lab):



1. Switch to root and verify:



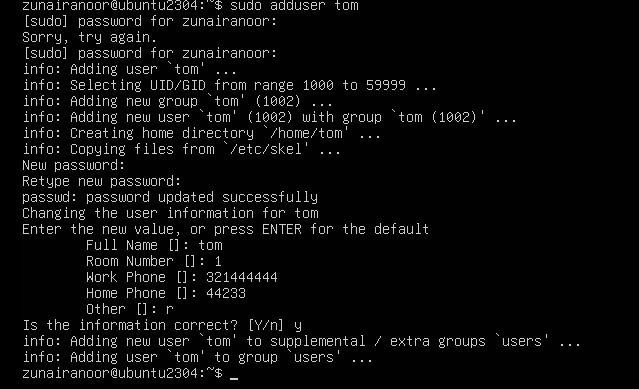
1. Switch back to your normal user:



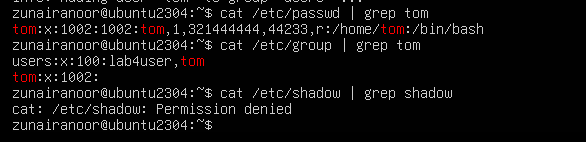
**Task 2 – Create user tom and verify in passwd/group/shadow**

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

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



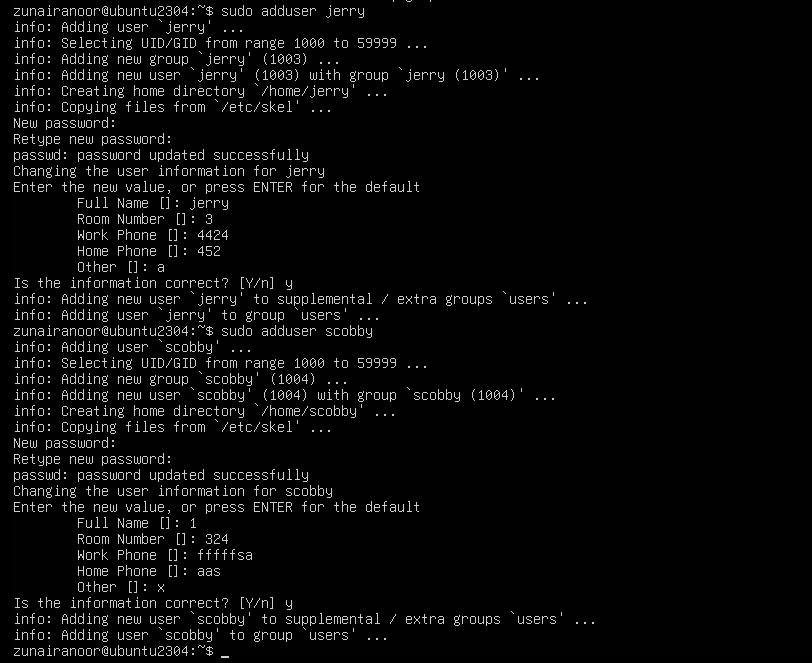
1. Verify tom in system files (view and visually confirm presence):



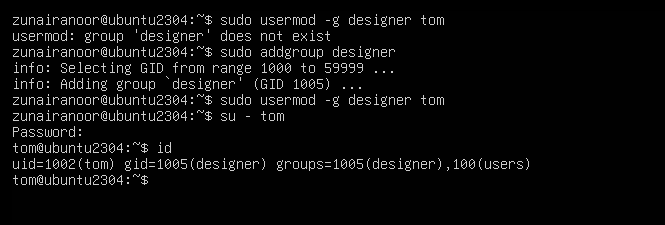
**Task 3 – Create groups; change tom’s primary and secondary groups**

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):



1. Change tom’s primary group to designer and verify:



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

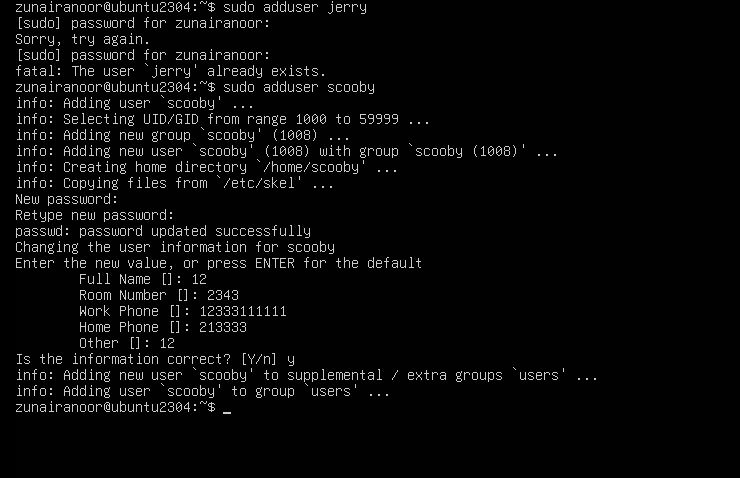


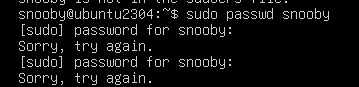
1. Replace all secondary groups so only tom (user’s own group) remains and verify:

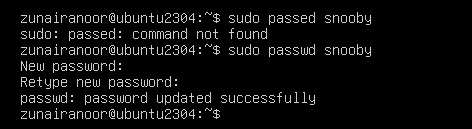


**Task 4 – Create/delete users (Jerry, Scooby) and groups (jolly, anime)**

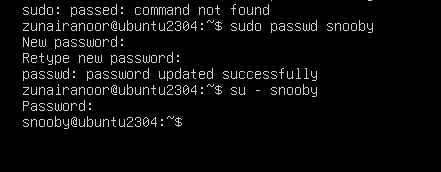
1. Create users:



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



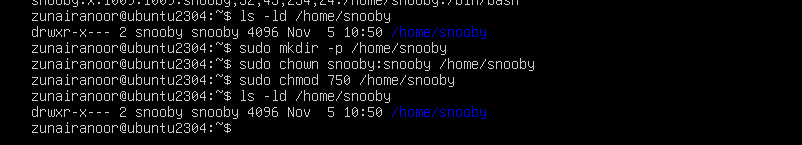
1. Try logging in as Scooby again (home directory still missing; expect a message such as “No directory, logging in with HOME=/”):



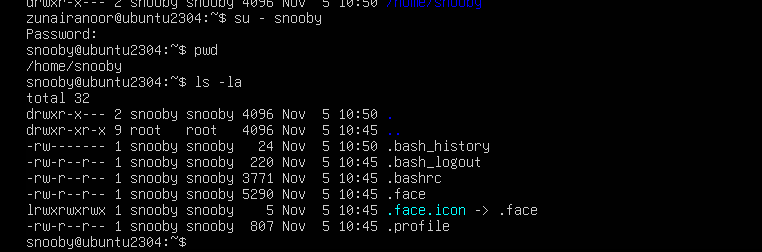
1. Show that Scooby’s home directory does not exist yet and what /etc/passwd says:



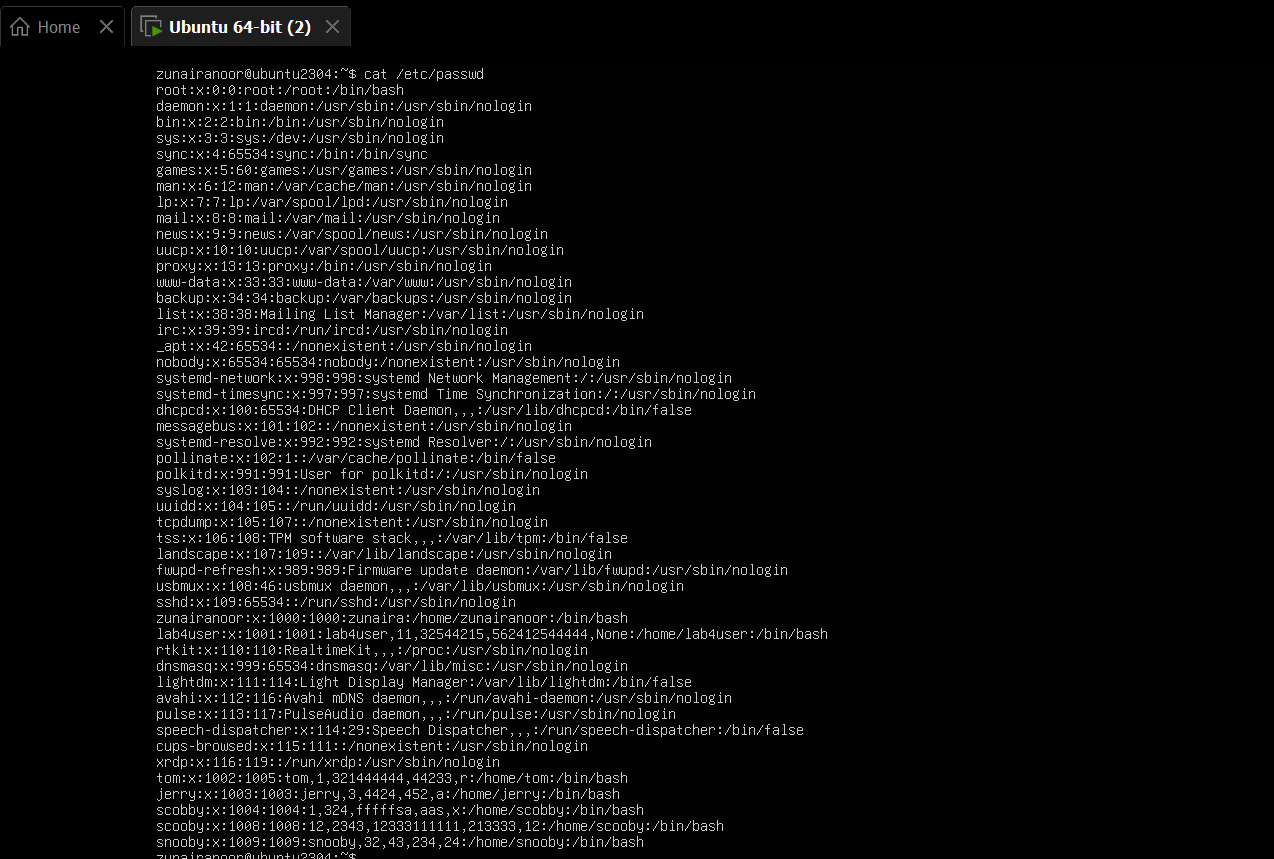
1. Manually create Scooby’s home directory and set proper ownership and permissions:



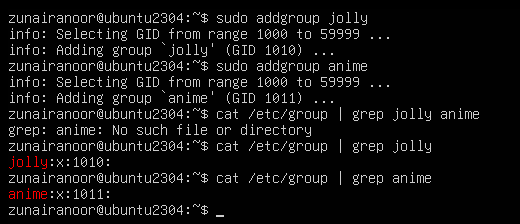
1. Log in as Scooby again and verify you land in the correct home directory:



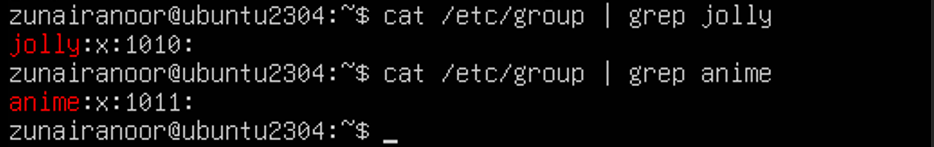
1. Verify users in system files and observe shell of Scooby:



1. Create groups:

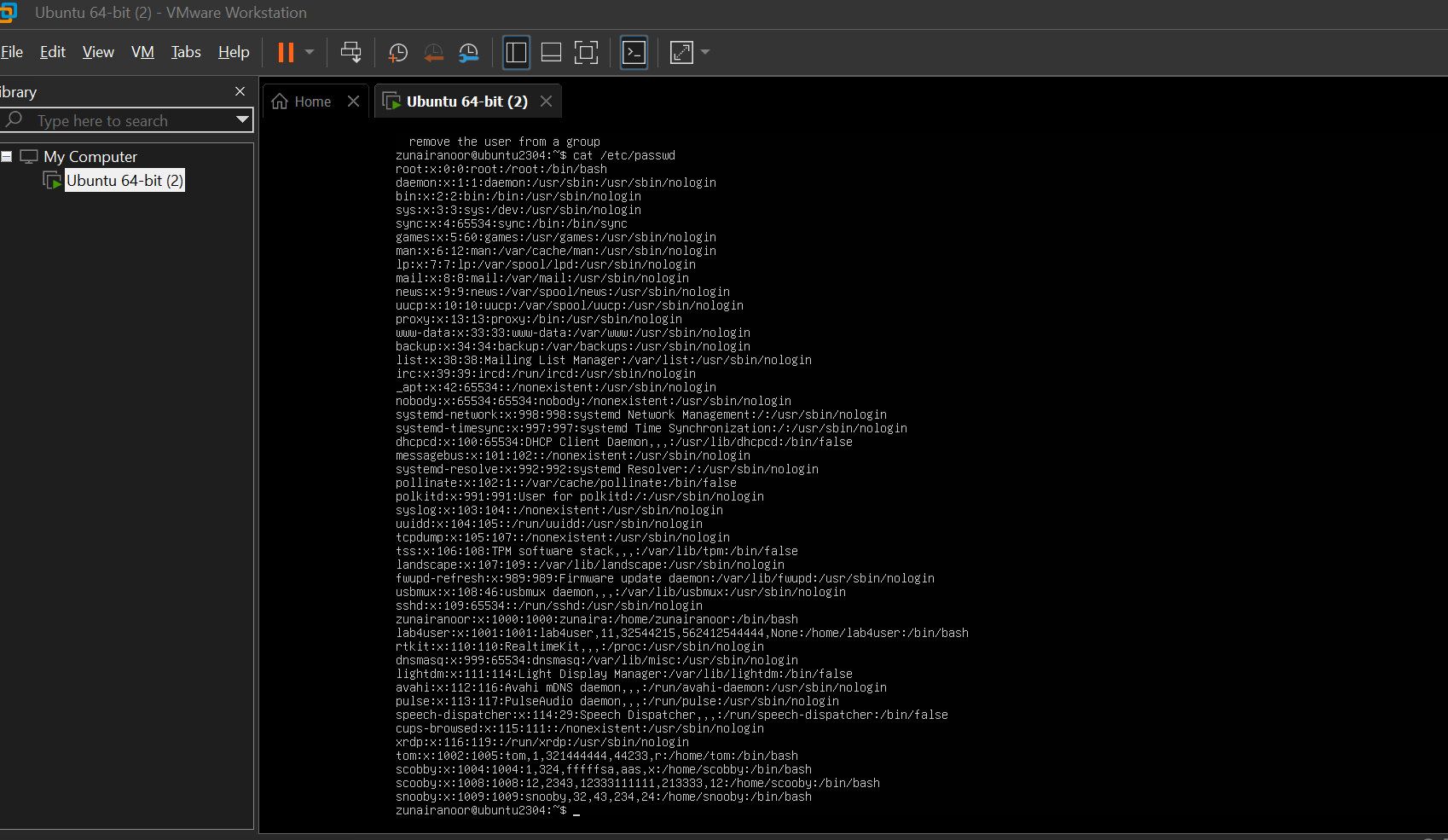


1. Verify groups:

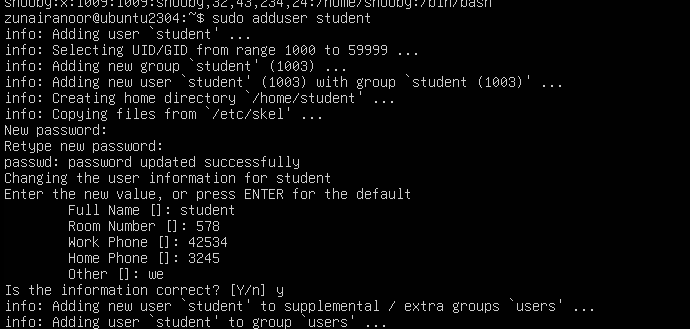


1. Delete groups and users:

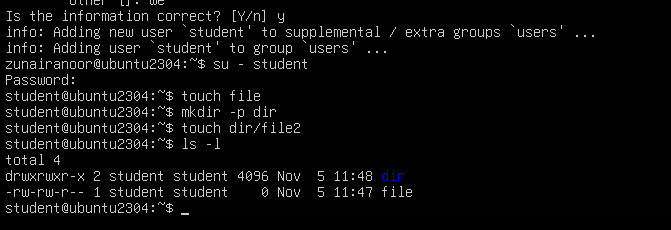




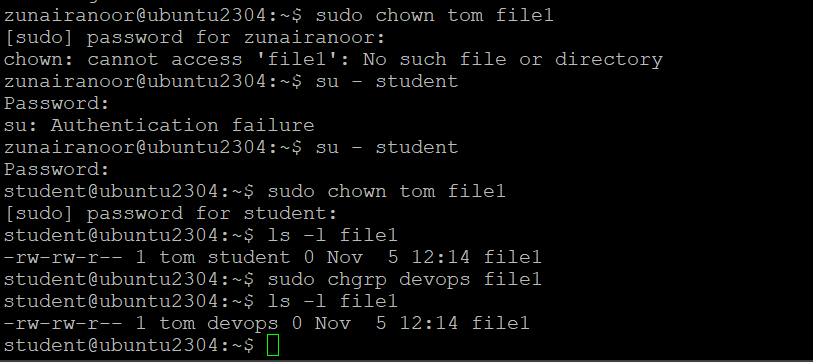
**Task 5 – Create user Student; create files; set owner/group; identify file types**



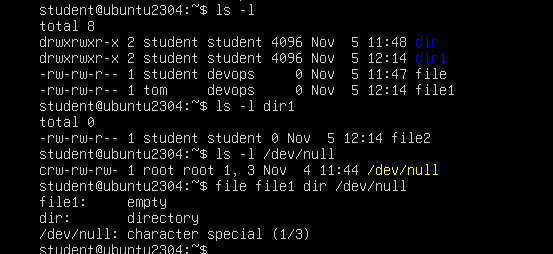
1. Switch to Student and create files:



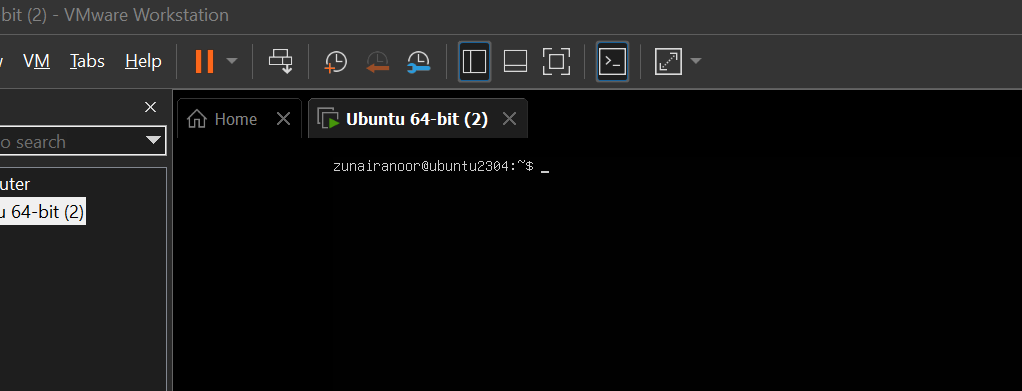
1. Change owner then group for file1 (separate commands):



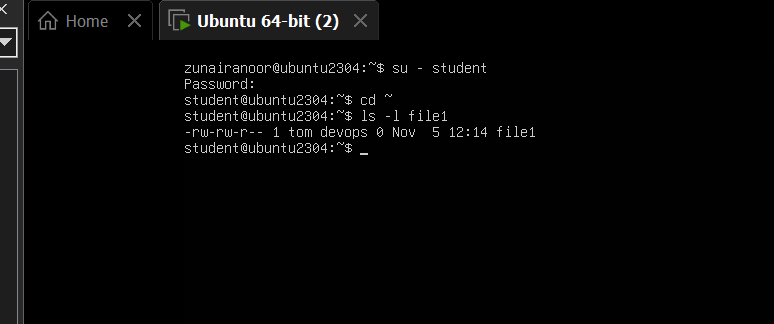
1. Identify files/directories and show /dev/null:



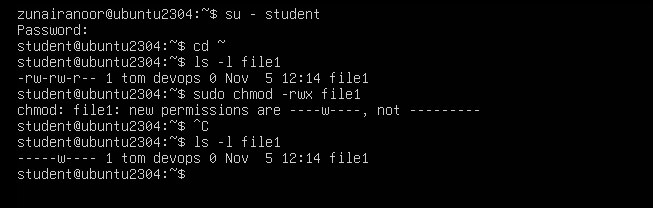
1. Exit Student:



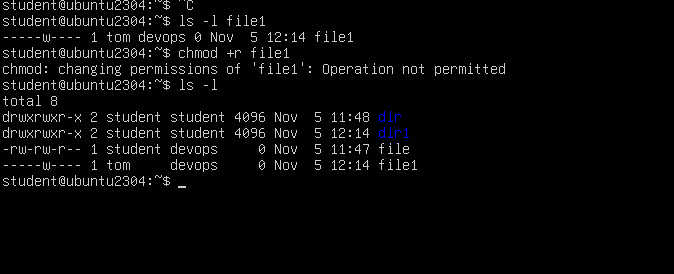
**Task 6 – Change permissions using symbolic mode**



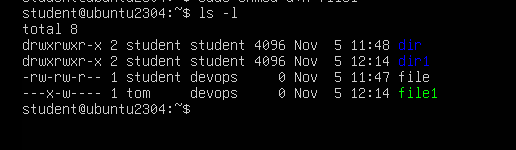
1. Remove all permissions:



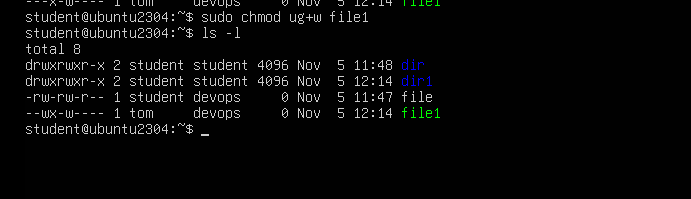
1. Add read to all:



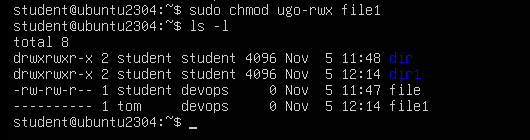
1. Add execute to user:

****

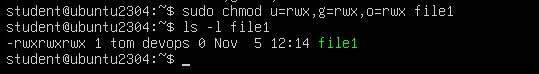
1. **Add write to user and group:**

****

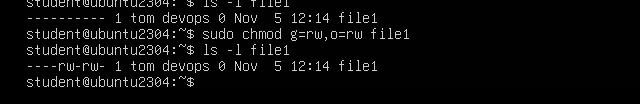
1. **Remove all permissions (explicit):**

****

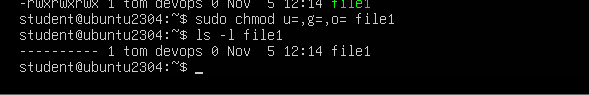
**Task 7 – Change permissions using “set” symbolic form (u= g= o=)**

****

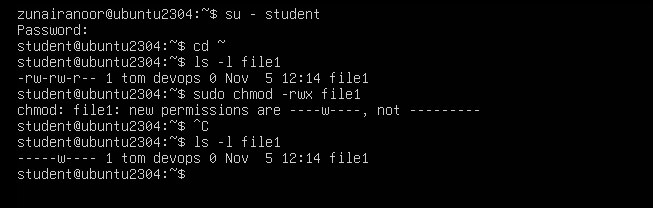
1. **Remove execute from group and others:**

****

1. **Remove all permissions:**

****

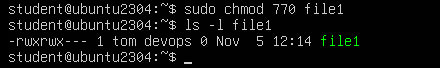
**Task 8 – Change permissions using numeric (octal) mode**



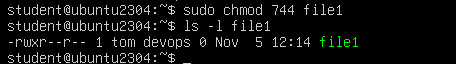
**chmod 777 file1**

****

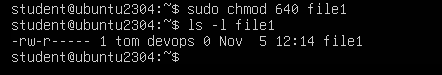
**chmod 700 file1**

****

**chmod 744 file1**

****

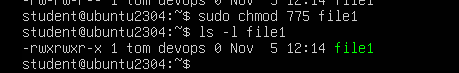
**chmod 640 file1**

****

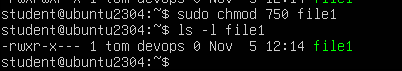
**chmod 664 file1**

****

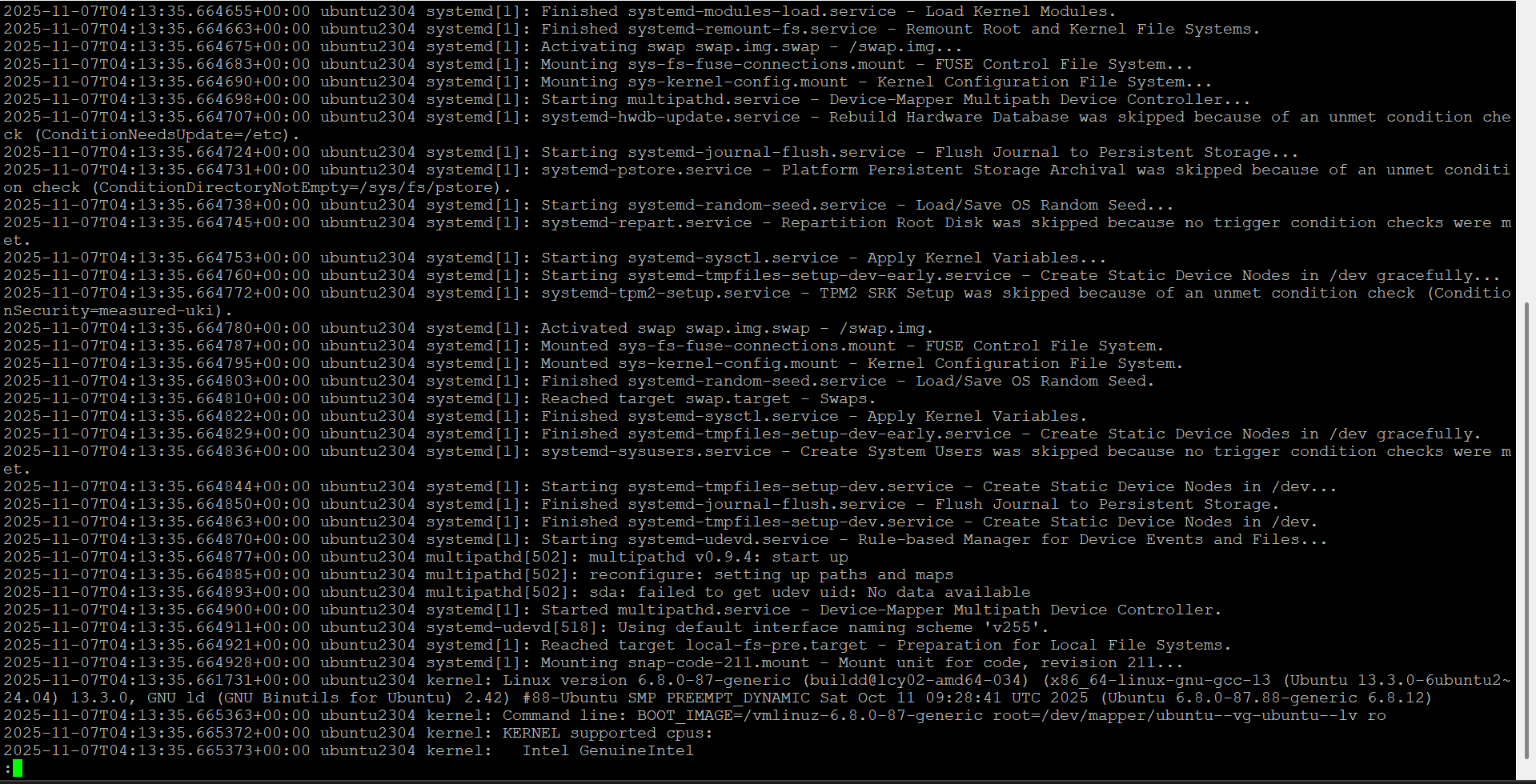
**chmod 775 file1**

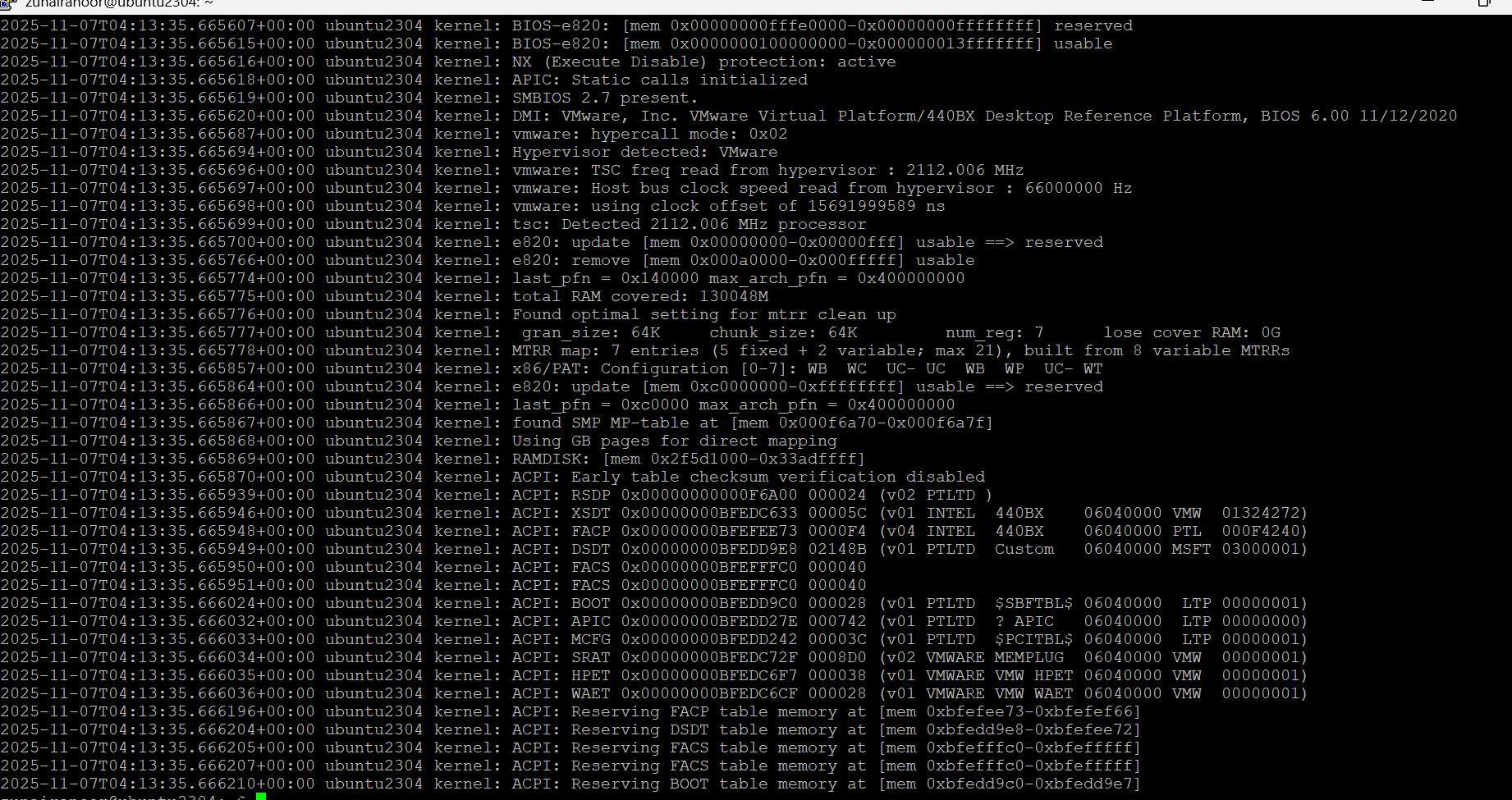
****

**chmod 750 file1**

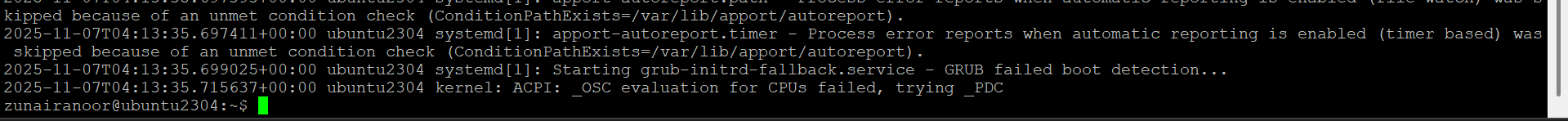
****

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



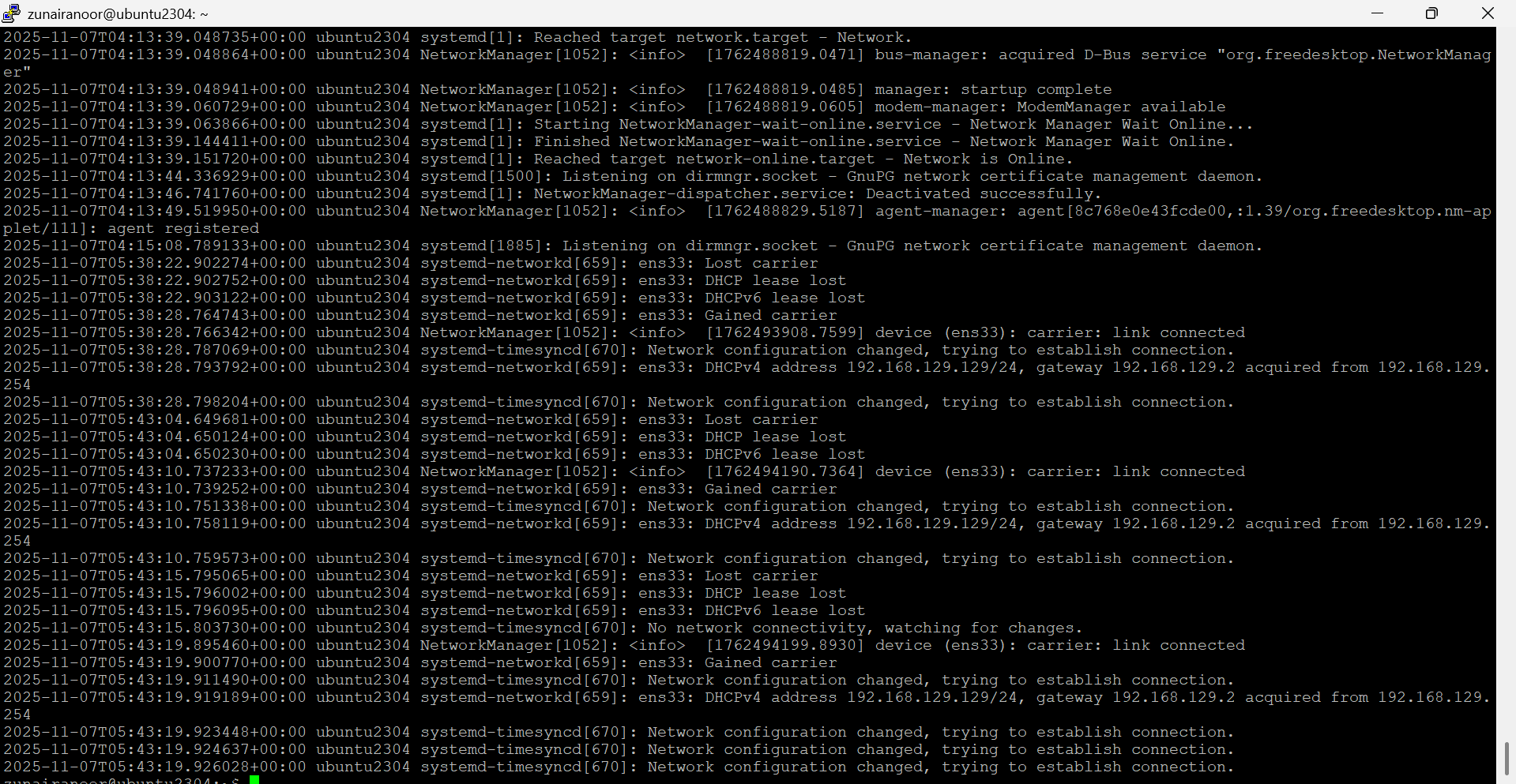


1. grep failures/errors:



1. redirect:

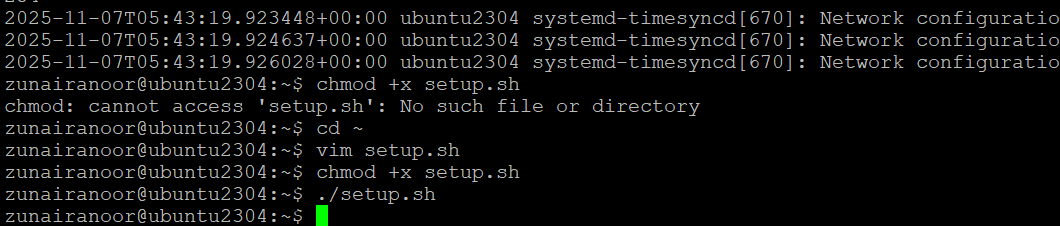




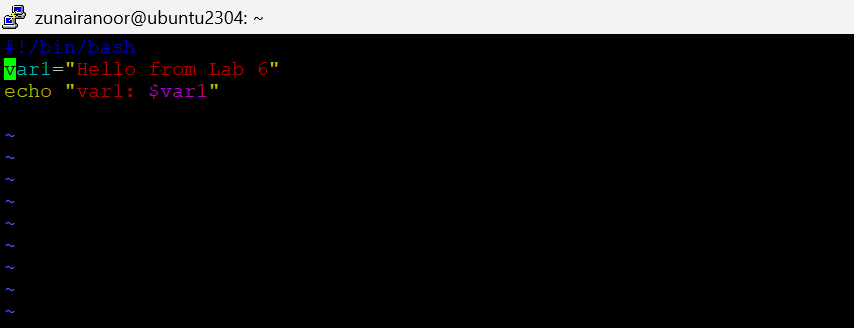
**Task 10 – Script setup.sh – variables, command substitution, file/dir checks, permissions (use vim)**

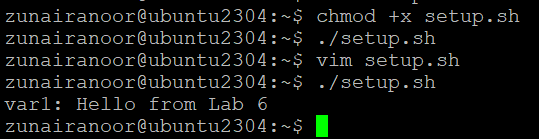
Make executable with:





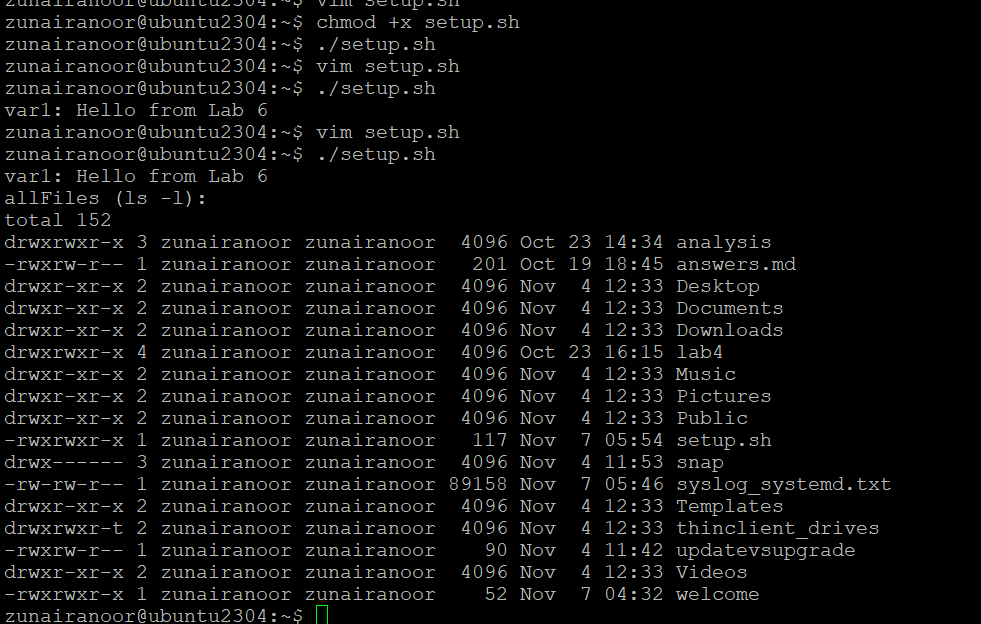
**Step 2 – Define variable var1 and echo it**



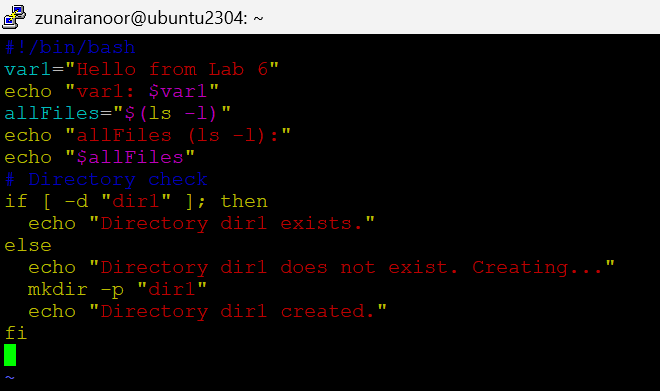


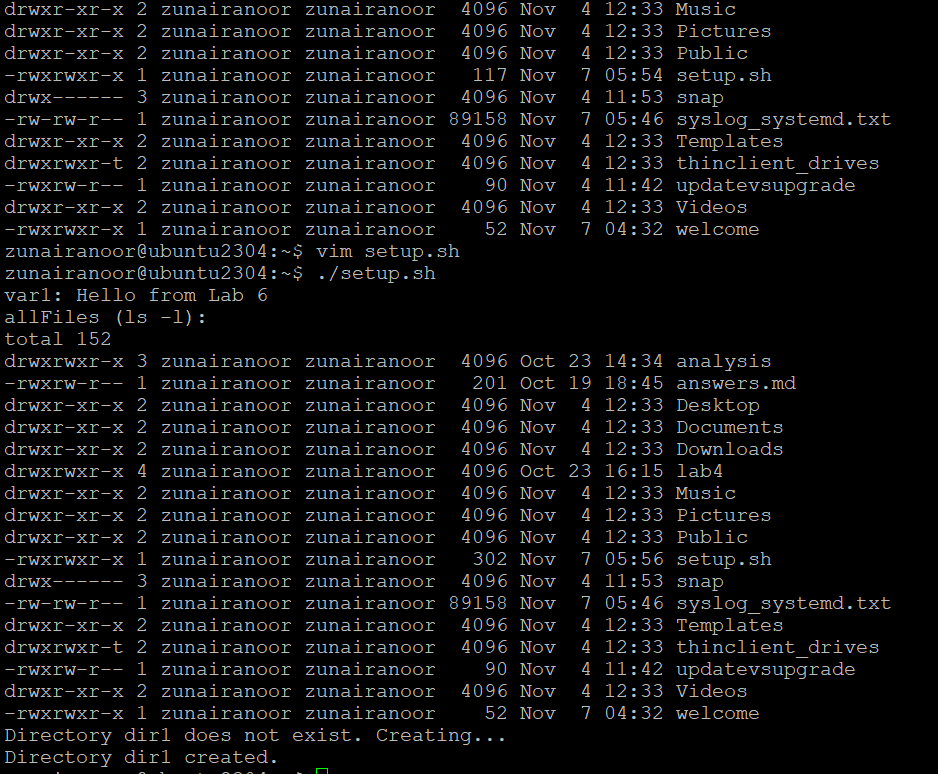
Step 3 – Save output of ls -l into variable allFiles and echo it



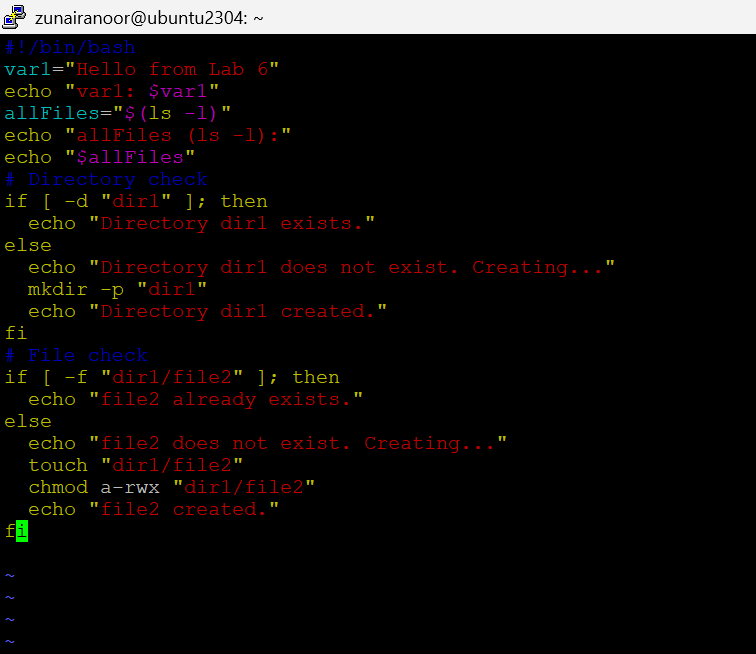


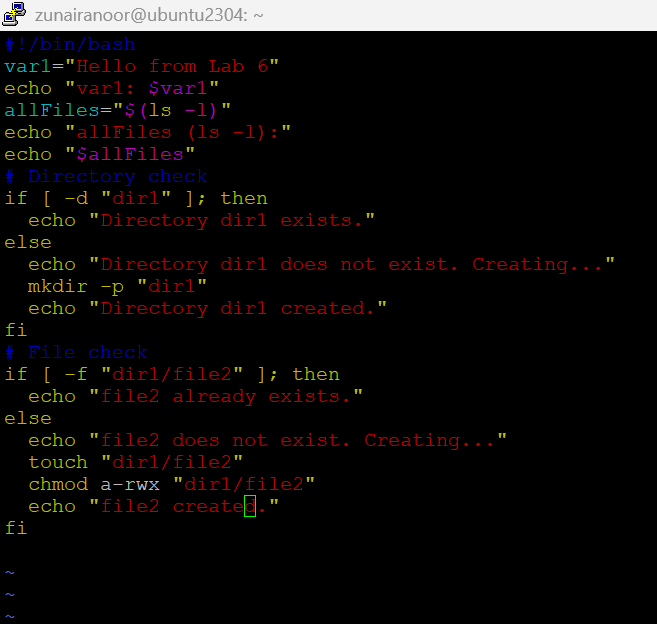
Step 4 – Directory check (dir1)

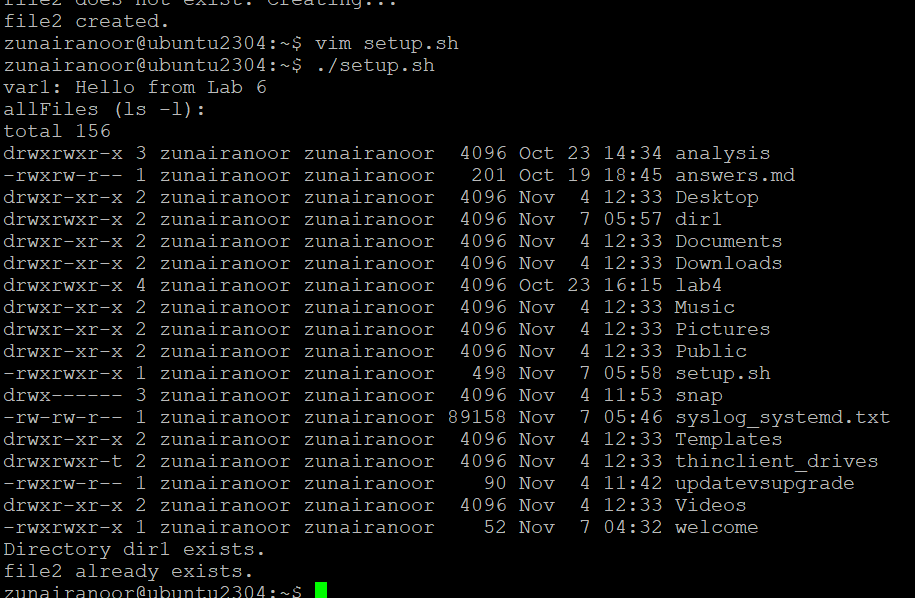




**Step 5 – File check (dir1/file2)**

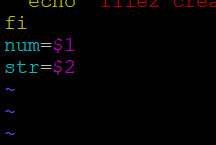


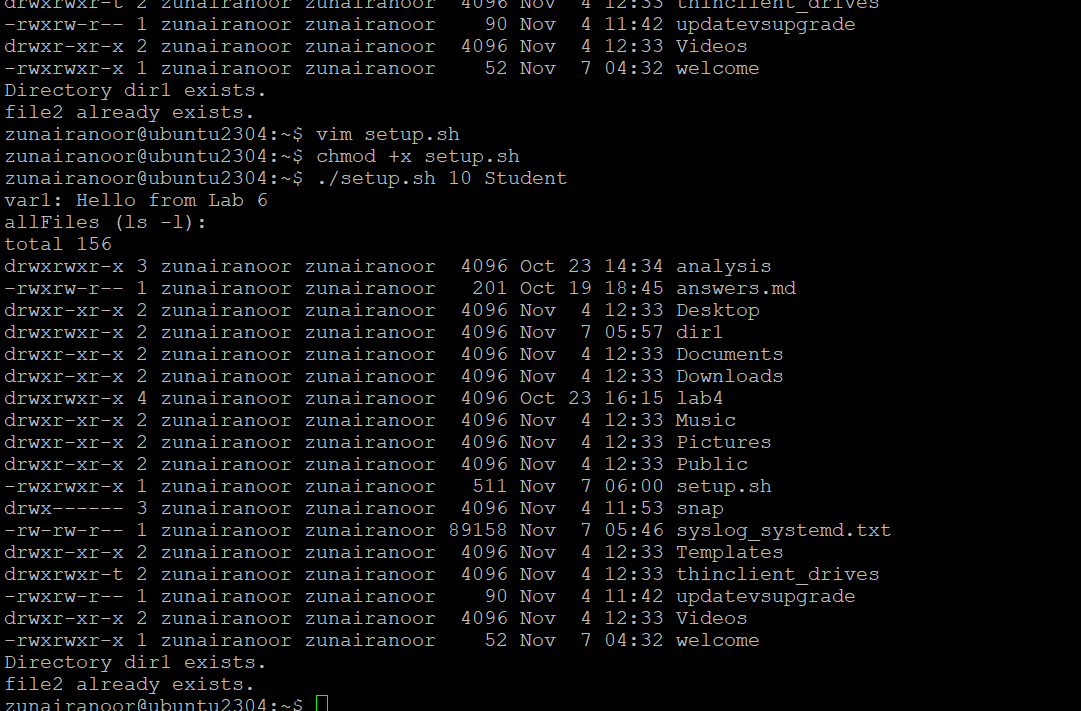
****

****

**Task 11 — Argument Comparisons**

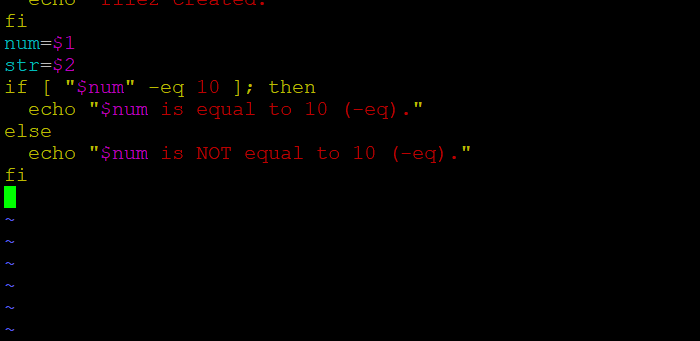
**🔹 Step 0 — Create file and define variables**

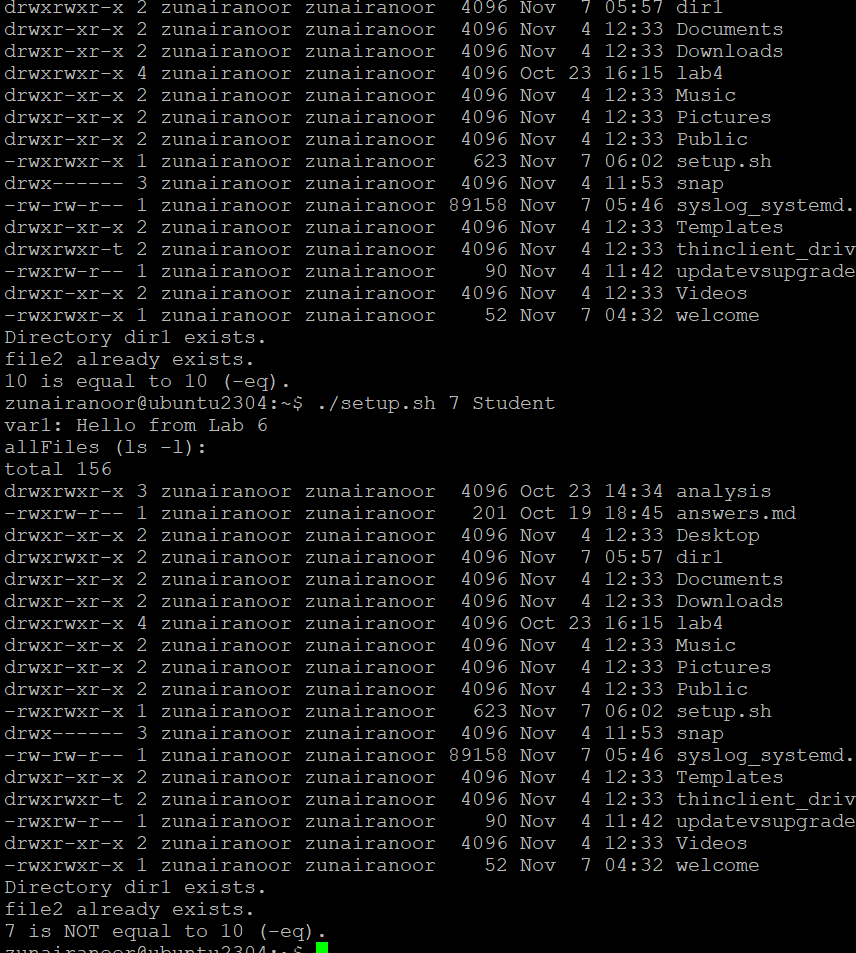
****

****

**🔹 Step 1 — -eq (equal)**

**Append:**

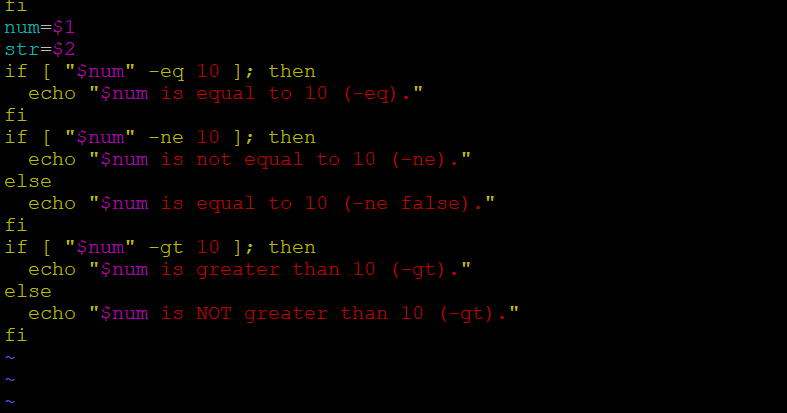
****

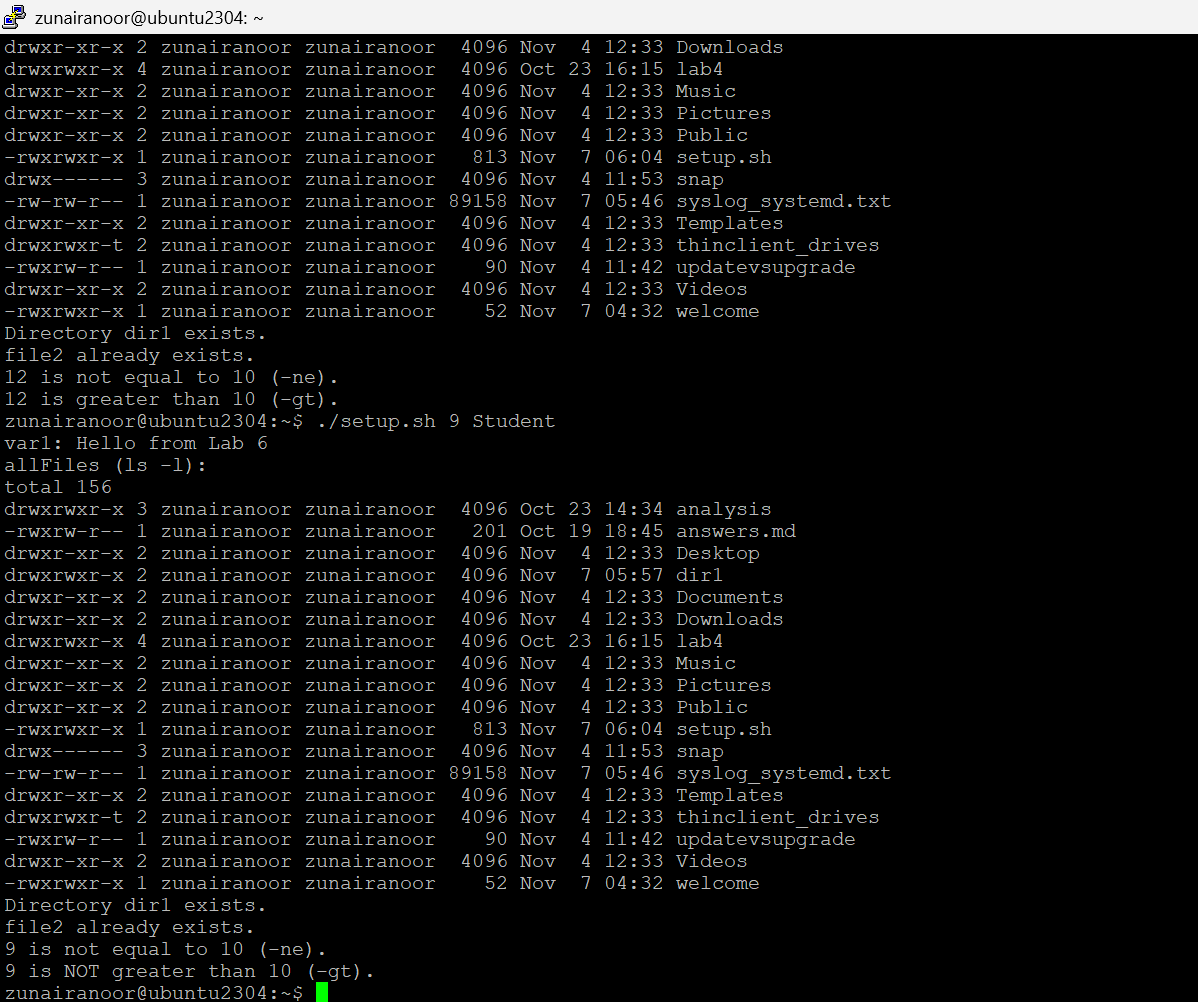
****

**Step 2 — -ne (not equal)**

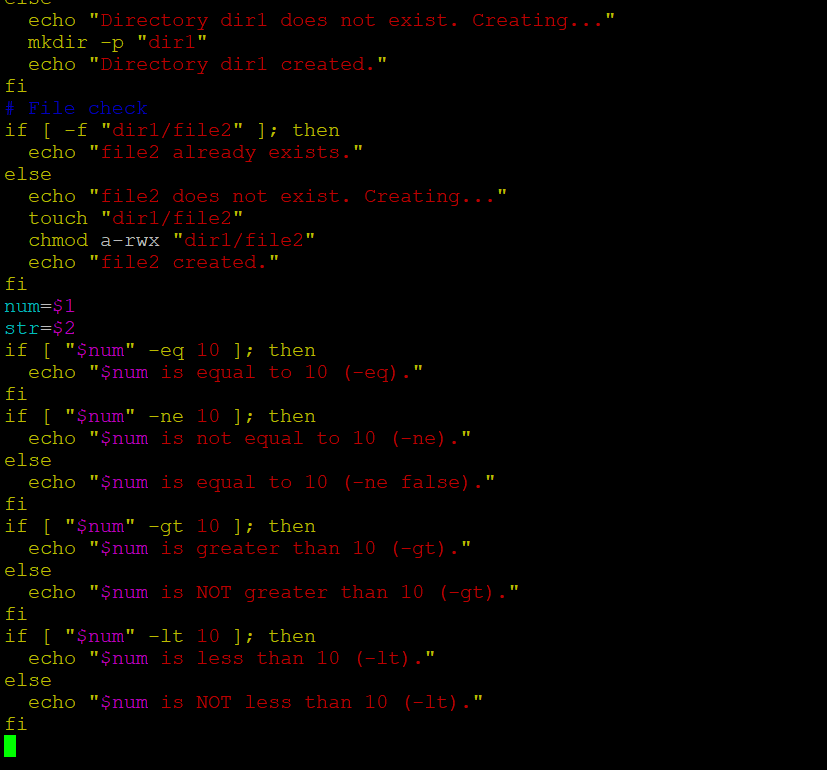
****

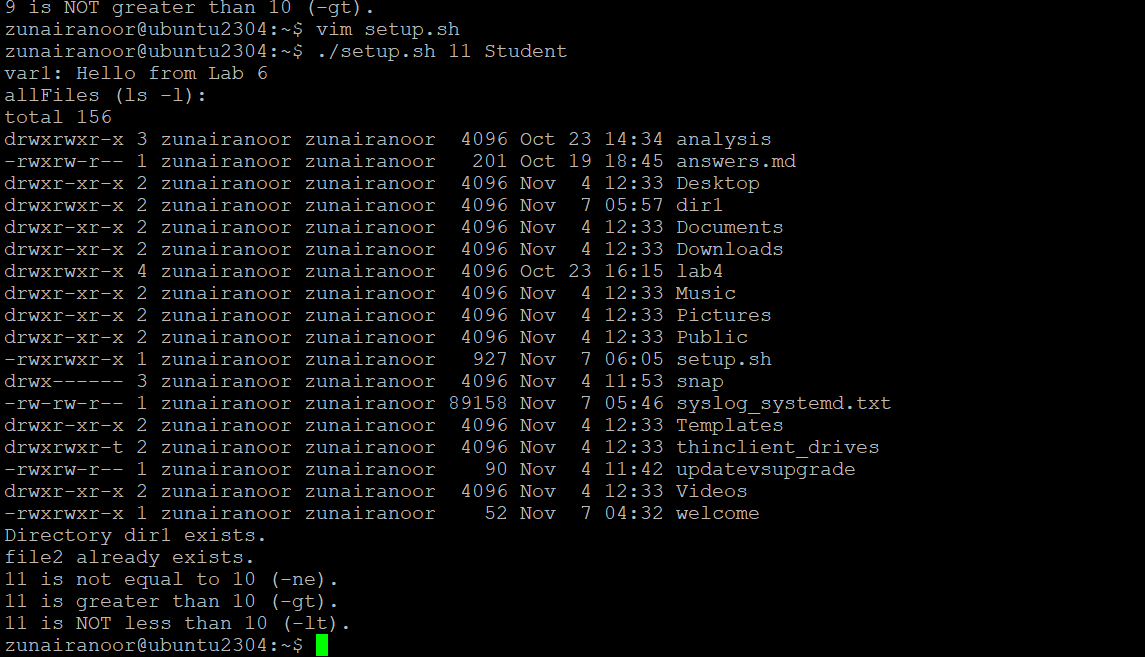
**Step 3 — -gt (greater than)**

****

****

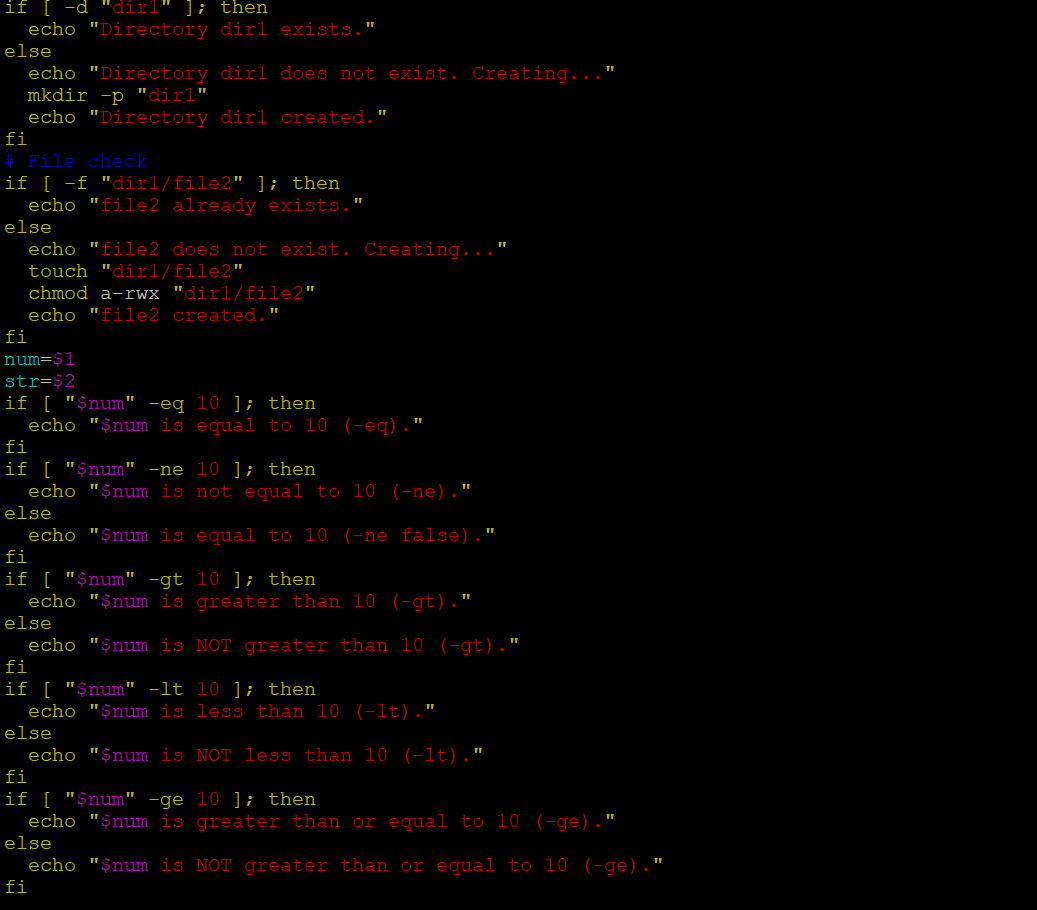
**🔹 Step 4 — -lt (less than)**

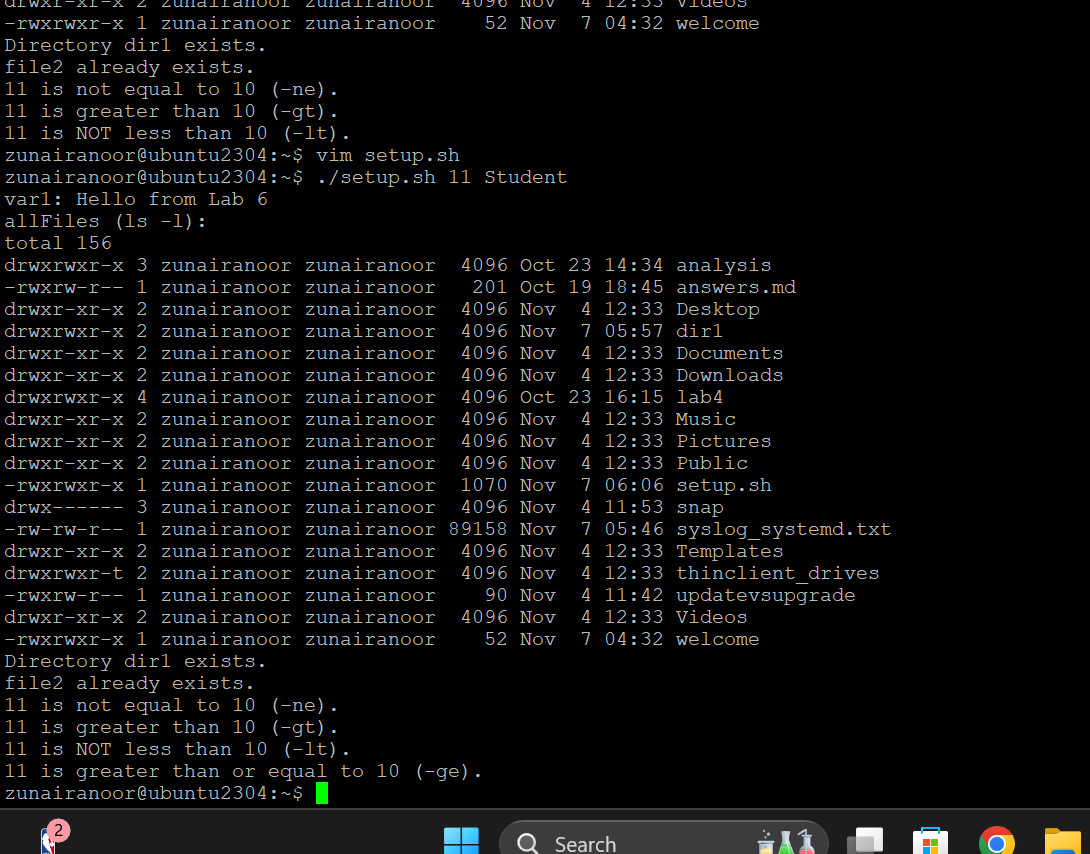
****

****

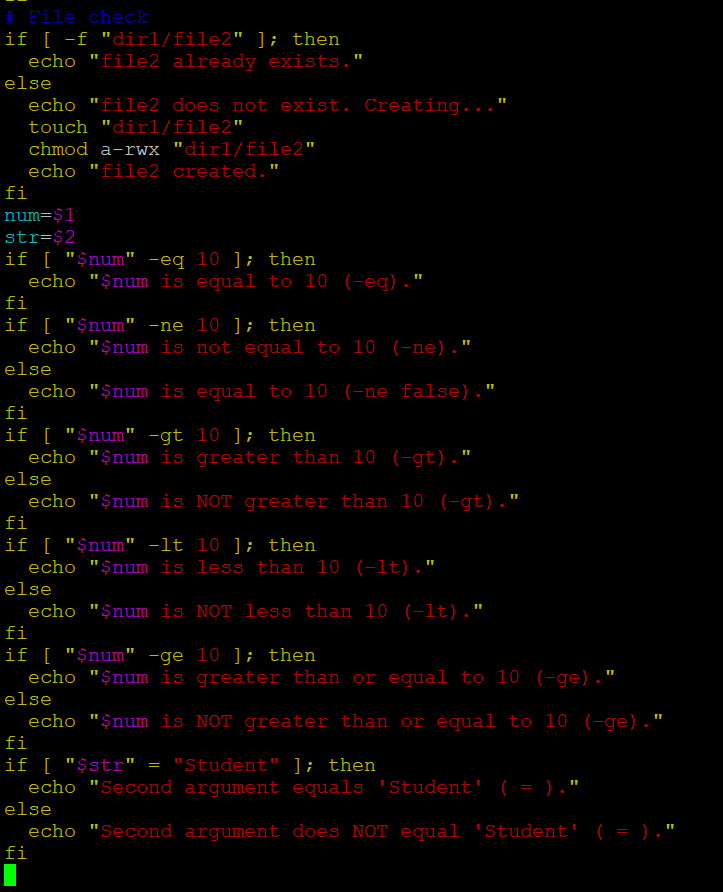
**Step 5 — -ge (greater or equal)**

**Append:**

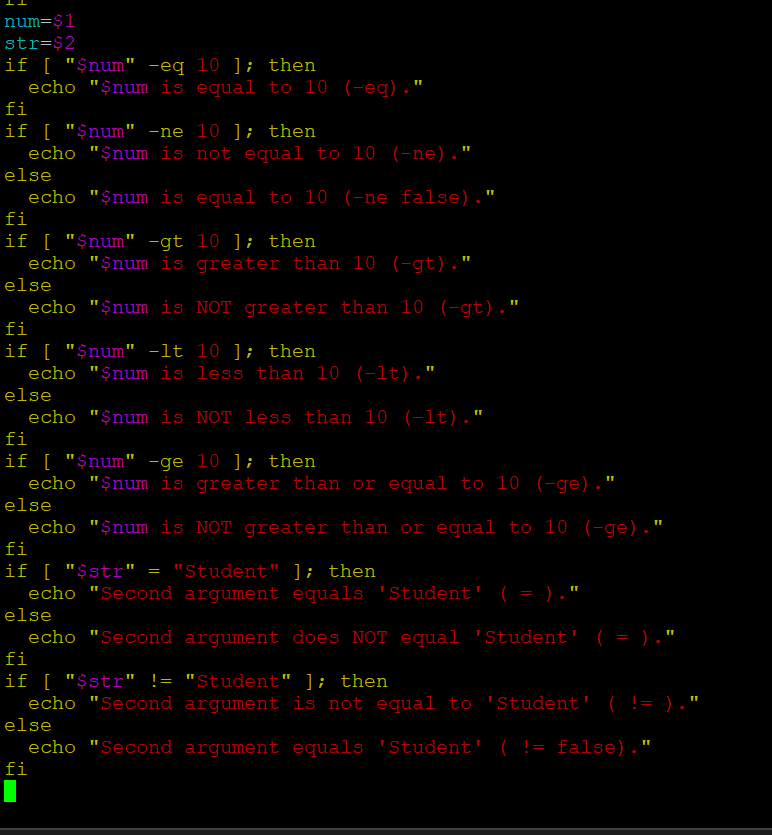
****

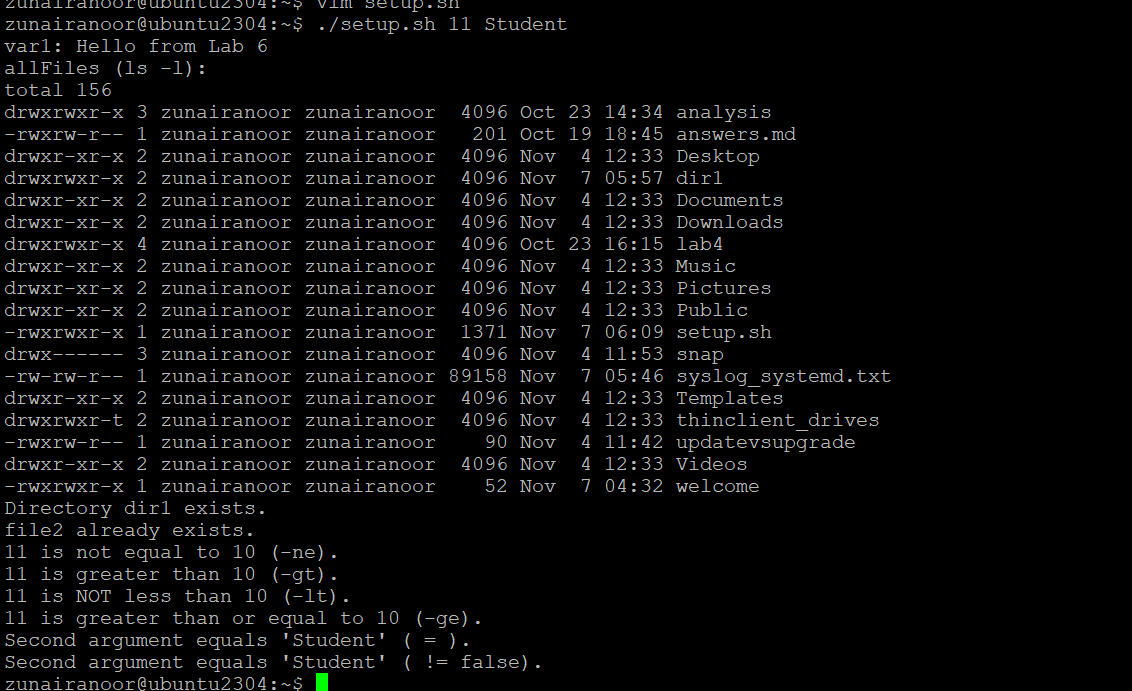
****

**Step 7 — String equality =**

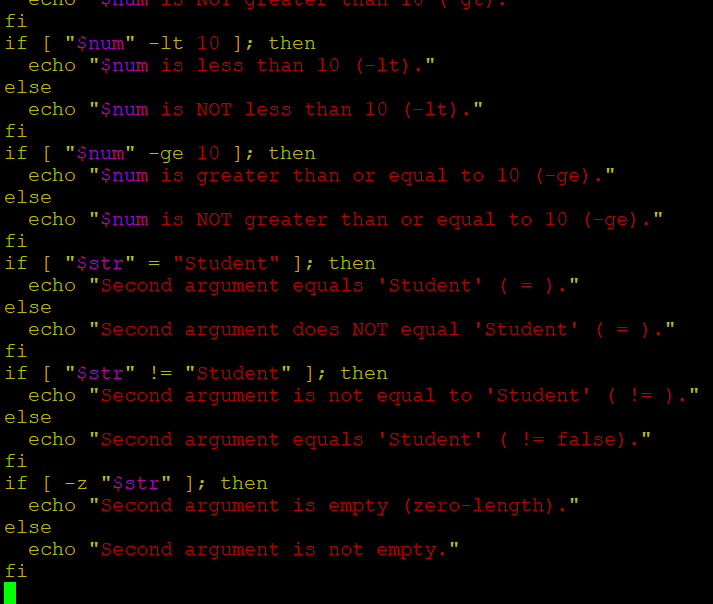
****

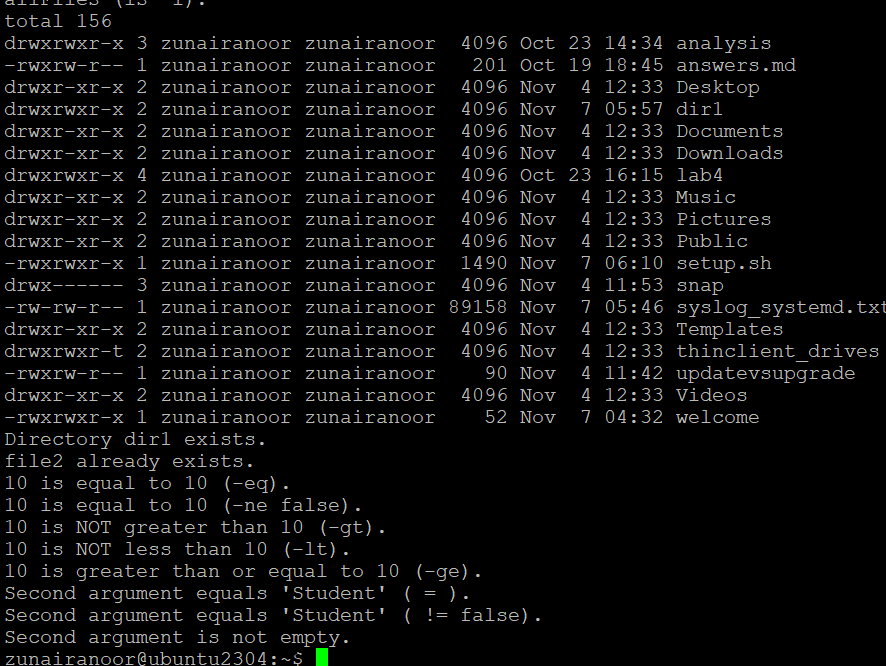
**Step 8 — String inequality !=**

****

****

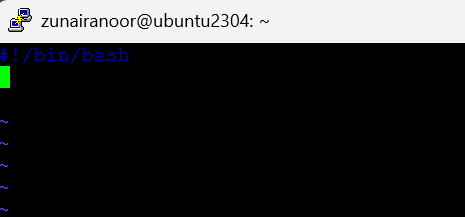
**Step 9 — Check empty string -z**

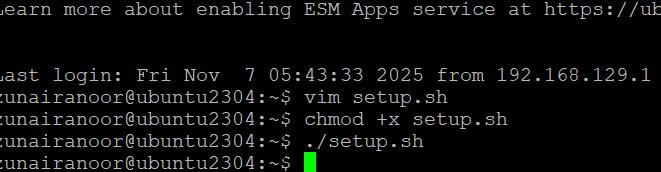
****

****

**Task 13 – setup.sh – While Loop Summation and Functions**

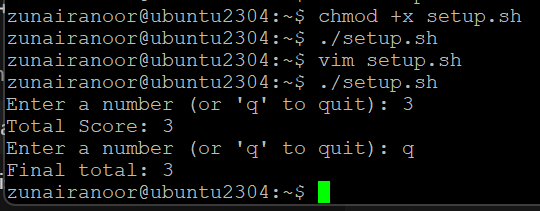
**🔹 Step 1 – Shebang only**

****

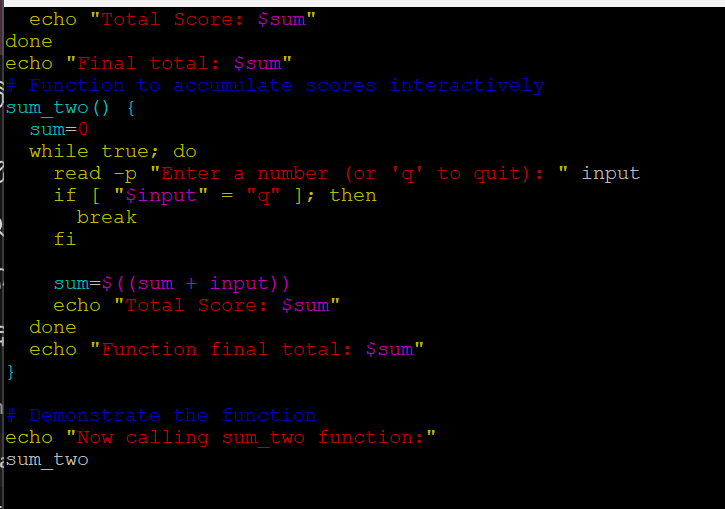
****

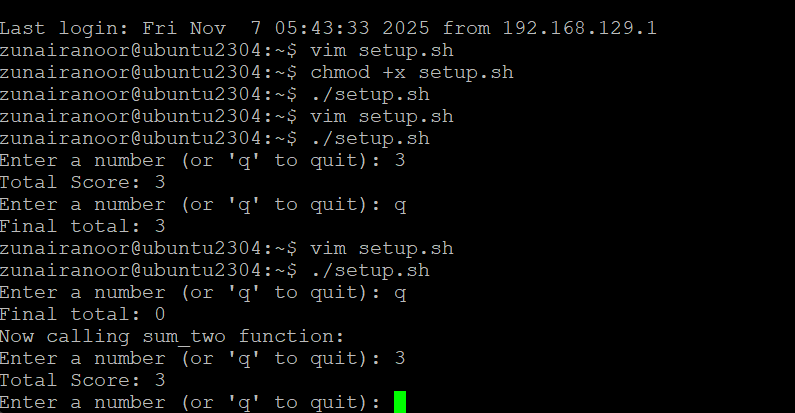
**Step 2 – Interactive While-Loop Summation**

****

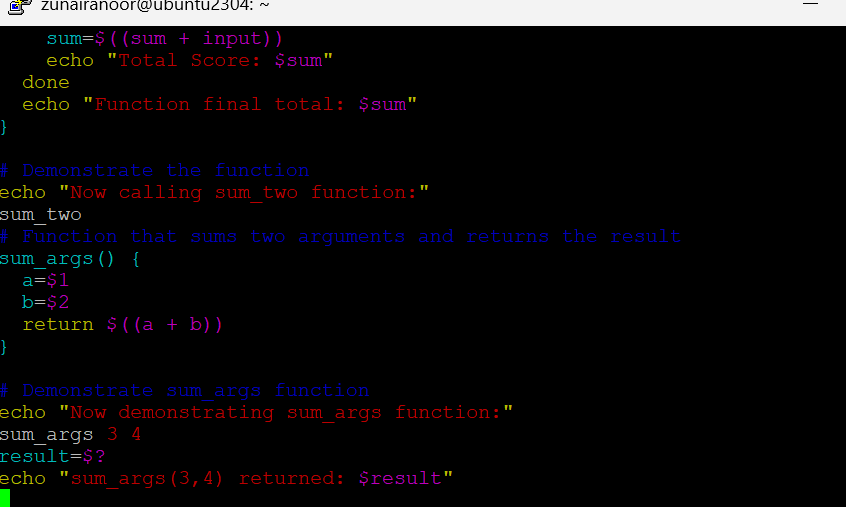
****

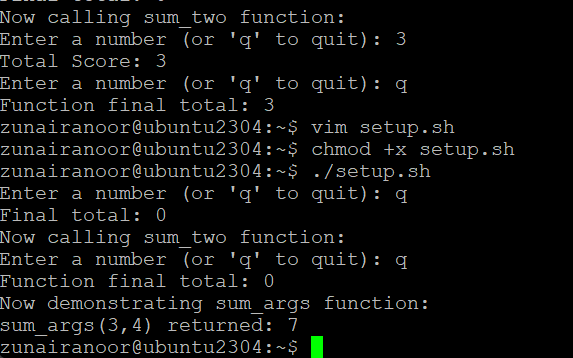
**Step 3 – Move into a Function (sum\_two)**

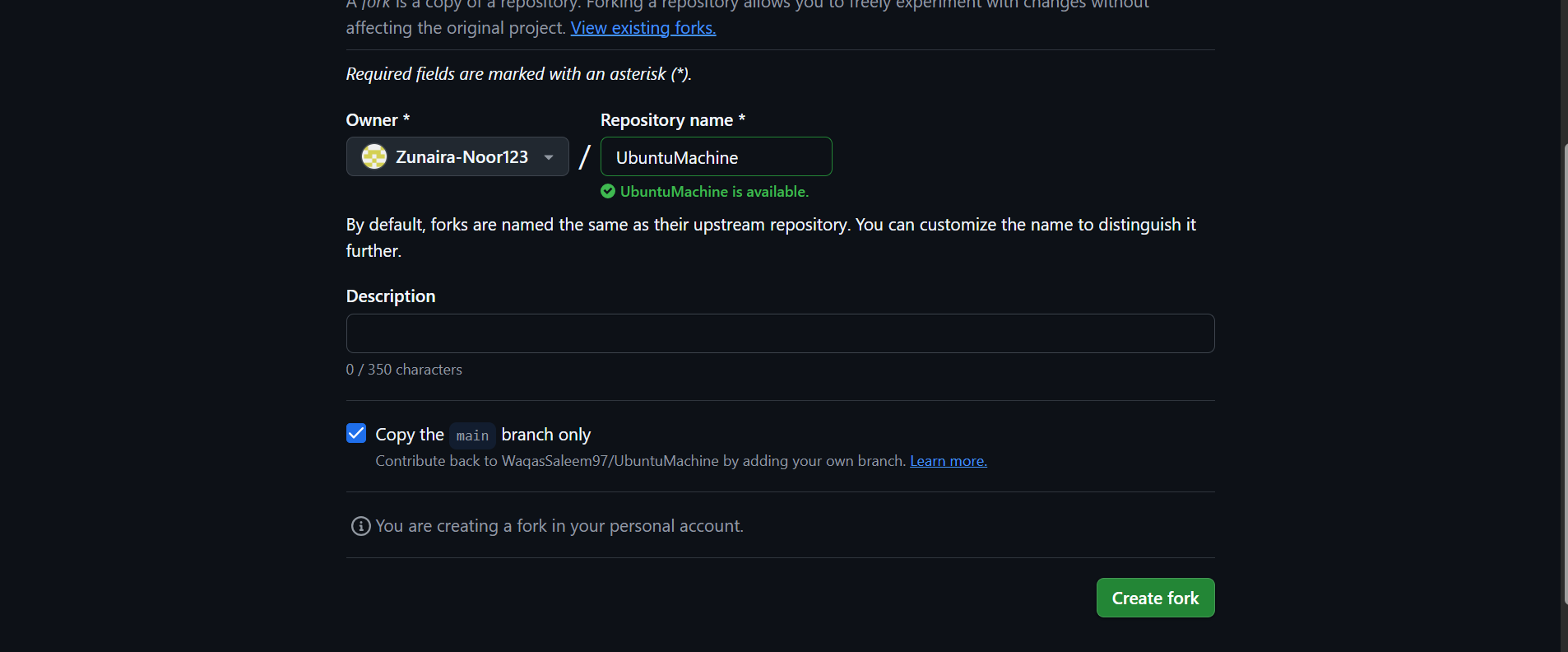
****

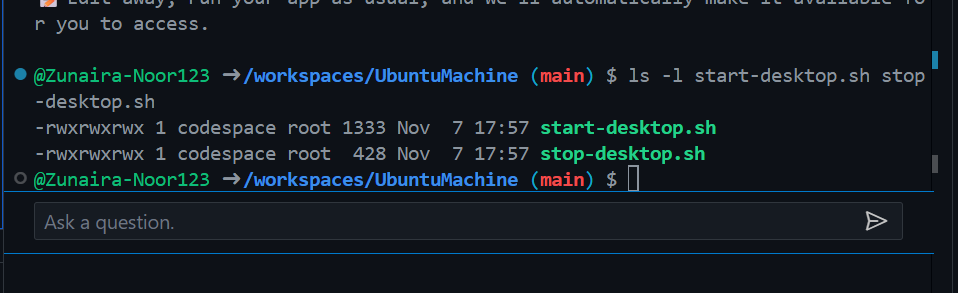
****

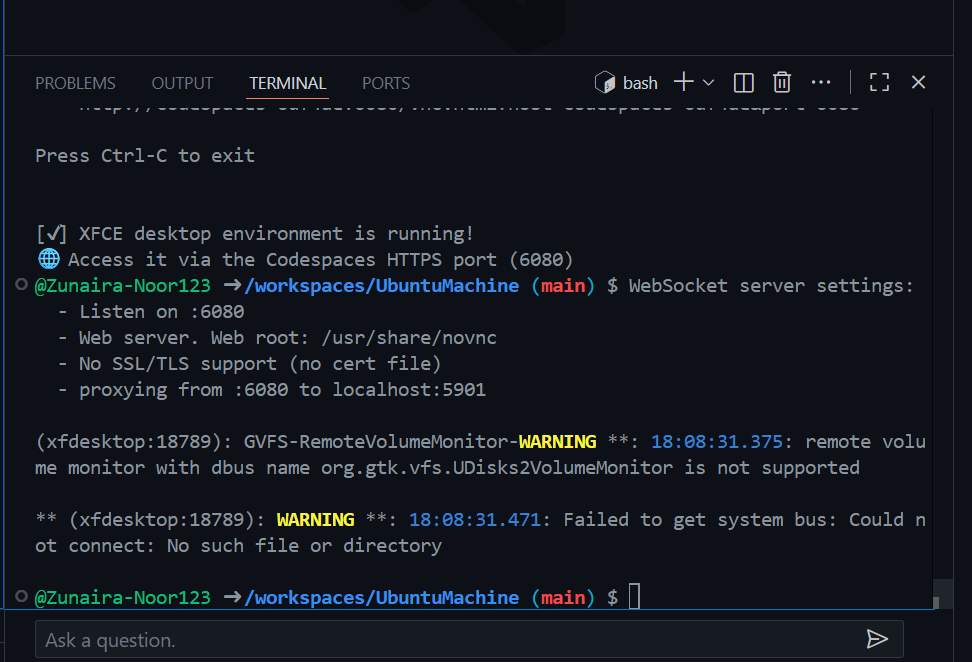
**Step 4 – Function with Two Numeric Arguments**

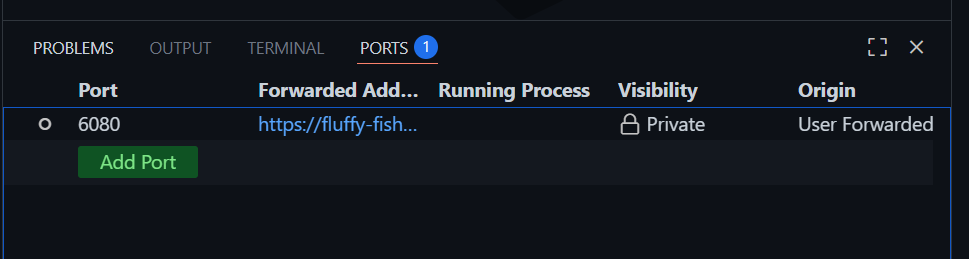
****

****

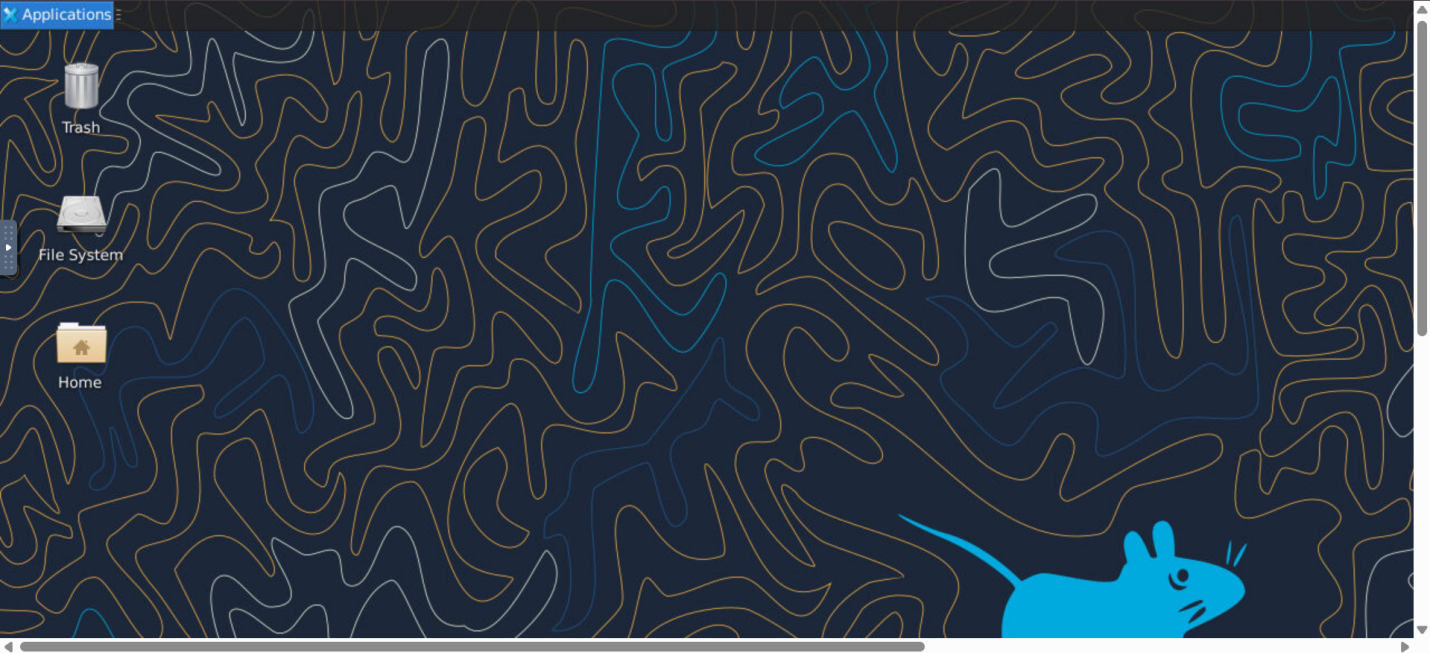
****

****

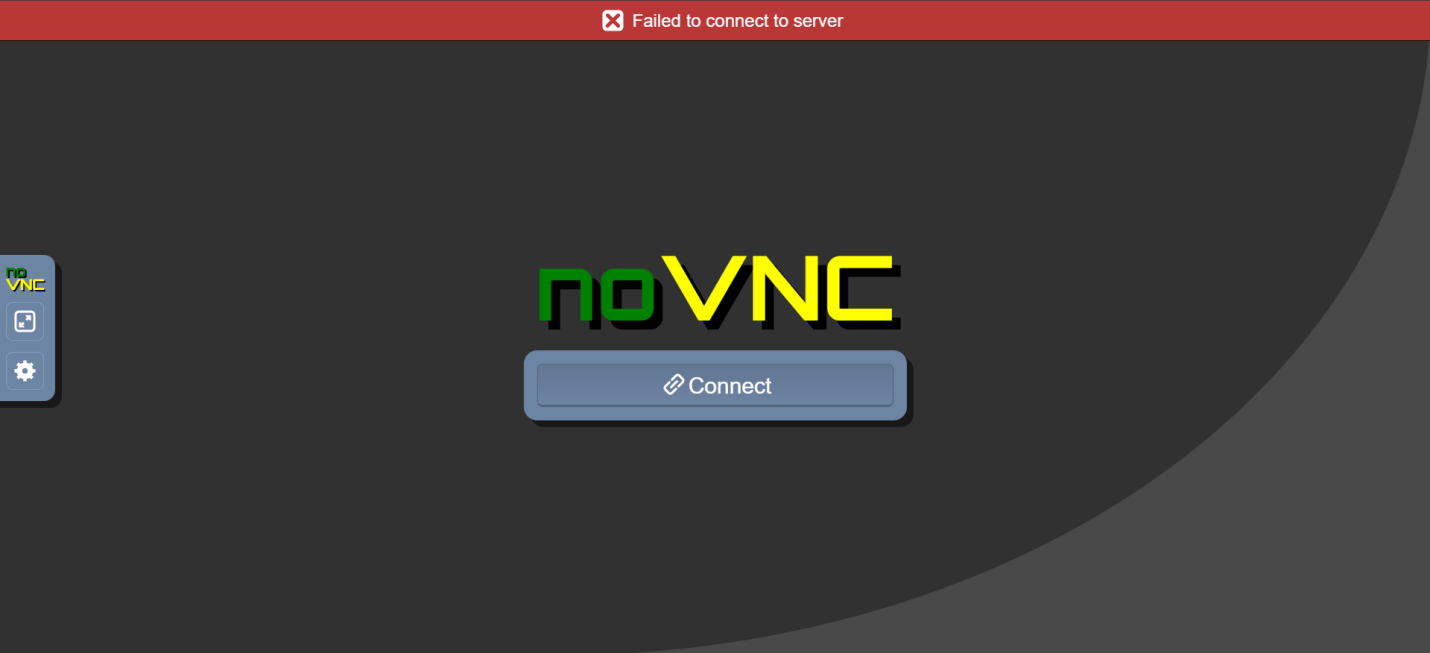
****

****

****

****

****

****