CSYE 6225: Network Structure & Cloud Computing Course

Tutorial and Practice # 1: Linux and Github

Objectives:
This tutorial covers basic Linux commands on Ubuntu, as well as a few GitHub commands for organizing your submissions.

Prerequisites:
You will need to create a micro Ubuntu instance on AWS, ssh to the created Ubuntu virtual machine (VM), and execute the Linux commands on this VM. Many of these commands, especially those related to user management, will not work on macOS. Ensure that you understand and run through all the commands on your Ubuntu VM.

Linux File System

Structure:

- **Root directory**: /
- **Key directories**:
  - /home/: User home directories.
  - /etc/: Configuration files.
  - /var/: Variable data (logs, caches).
  - /bin/: Essential binary commands.
  - /usr/: User binaries, libraries, documentation, and source code.

Navigating the File System

pwd: Print working directory (current location).
ls: List directory contents.

- -l: Long listing format.
- -a: Include hidden files.

```
ls -la
```

cd: Change directory.

```
cd /path/to/directory
```

Shortcut: cd ~ moves to the home directory.

tree: Show directory structure (install via package manager).
```
sudo apt update
sudo apt install tree
tree /home
```

File Operations

Creating and Managing Files:

touch: Create an empty file.
```
touch newfile.txt
```
cat: Display file contents.
```
cat newfile.txt
```
nano/vim: Edit files using text editors.
```
nano newfile.txt
```
cp: Copy files or directories.
```
cp source.txt destination.txt
```
mv: Move or rename files.
```
mv oldfile.txt newfile.txt
```
rm: Delete files (be careful with this command).
```
rm newfile.txt
```

- -r: Remove directories and their contents recursively.

```
mkdir ~/myfolder
ls ~/myfolder
rm -r ~/myfolder/
```

Creating and Managing Directories:

mkdir: Create a new directory.
```
mkdir mydirectory
```
rmdir: Remove empty directories.
```
rmdir mydirectory
```

Managing File Permissions

Permission Overview: Each file and directory has three sets of permissions (owner, group, others):

- **r**: Read.
- **w**: Write.
- **x**: Execute.

Commands:

ls -l: View file permissions.

Output format: `-rwxr-xr-- 1 user group 4096 Jan 1 12:00 filename.`

chmod: Change file permissions.

```
chmod 755 filename
```

Example: 755 gives the owner full permissions and read/execute permissions to others.

chown: Change file owner and group.

```
sudo chown user:group filename
```

Managing Processes

Commands:

ps: Display running processes.

```
ps aux
```

top/htop: Interactive process viewer (install htop using the package manager).

```
top
```

To install htop (if undefined):

```
sudo apt-get install htop
htop
```

kill: Terminate a process.

```
kill PID
```

killall: Terminate all instances of a process by name.

```
killall process_name
```

Networking and System Information

Commands:

ifconfig/ip addr: View network interfaces and IP addresses.

```
ip addr
```

ping: Check network connectivity.

```
ping google.com
```

df: Check disk space usage.

```
df -h
```

du: Estimate file space usage.

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

Simple Bash Script

Script Example: Automate a task such as backing up a folder.

Create a script: Use nano or vim.

```
nano backup.sh
```

Script contents:

```
#!/bin/bash
# Backup script

SRC="/path/to/source"
DEST="/path/to/backup/destination"

tar -czf $DEST/backup-$(date +%Y-%m-%d).tar.gz $SRC
echo "Backup completed!"
```

Make the script executable:

```
chmod +x backup.sh
```

Run the script:

```
./backup.sh
```

h. User Management

Creating Users:

useradd: Add new users to the system.

```
sudo useradd -m user1
sudo useradd -m user2
sudo useradd -m user3
```

Flag: -m ensures the creation of a home directory for the user.

Setting Passwords:

passwd: Change or set passwords for users.

```
sudo passwd user1
sudo passwd user2
sudo passwd user3
```

Note: You will be prompted to enter and confirm the password for each user.

Switching Users:

su: Switch to another user account.

```
su - user1
```

Enter the password for user1 when prompted. The - flag ensures that the user's environment is loaded.

Exit back to the original user:

```
exit
```

Verifying Users:

who: Check which users are logged into the system.

```
who
```

Git: Amend, Reset, and Squashing.

Amending the Last Commit

If you need to change the most recent commit (either to modify the commit message or add new changes), you can use the --amend option.

Modify Files (under a git repository). You can clone any repository on your Ubuntu VM

Make the necessary changes to your files. For example, edit a file:

```
nano file.txt
```

Stage the Changes

After modifying the file, stage the changes using git add:

```
git add file.txt
```

Amend the Commit

Now, amend the last commit:

```
git commit --amend
```

If you only want to change the commit message, you can run the following:

```
git commit --amend -m "New commit message"
```

This will keep the changes the same but update the commit message.

Squashing Commits

Squashing allows you to combine multiple commits into one. This is useful for cleaning up your Git history before merging.

Step 1: View Commit History

Check your recent commits to decide how many commits you want to squash:

```
git log --oneline
```

You will see something like this:

```
abcd123 Second commit
efgh456 First commit
```

Step 2: Start an Interactive Rebase

Use the interactive rebase command to squash commits:

```
git rebase -i HEAD~n
```

Replace `n` with the number of commits you want to squash. For example, if you want to squash the last 2 commits, use:

```
git rebase -i HEAD~2
```

Step 3: Mark Commits to Squash

In the text editor that opens, you will see a list of commits like this:

```
pick abcd123 Second commit
pick efgh456 First commit
```

Change pick to squash (or s for short) for the commits you want to squash. Only the first commit should remain as pick. For example:

```
pick efgh456 First commit
squash abcd123 Second commit
```

Save and close the file.

Step 4: Edit the Commit Message

Git will combine the commit messages, and you will be prompted to edit them. You can either keep the combined messages or modify them as needed. After editing, save and close the editor.

Step 5: Push the Squashed Commit

If you already pushed the previous commits to a remote repository, you'll need to force-push after squashing:

```
git push --force
```

Reset a Specific File Locally

To reset a file in your local repository to its state in the last commit or a specific commit:

Step 1: Reset the File Locally

You can reset a specific file to its state in the most recent commit:

```
git checkout HEAD -- path/to/file.txt
```

- This will discard any local changes made to file.txt and revert it to its last committed state.

If you want to reset the file to a specific commit, replace HEAD with the commit hash:

```
git checkout <commit_hash> -- path/to/file.txt
```

Step 2: Stage the Reset File

After resetting the file, you need to stage it so that the reset is included in the next commit:

```
git add path/to/file.txt
```

2. Commit the Changes Locally

Now commit the reset file locally to finalize the changes:

```
git commit -m "Reset file.txt to the last commit"
```

3. Push the Changes to the Remote Repository

To update the remote repository with your reset changes, push them to the remote:

```
git push origin <branch_name>
```

Replace <branch_name> with the name of the branch you're working on (e.g., main, master, feature).

Hard Reset to Remote (Force Push)

If you need to reset an entire branch and force the remote repository to match your local state (including overwriting changes in the remote):

Step 1: Hard Reset Locally

You can reset the branch to a specific commit or HEAD (the latest commit):

```
git reset --hard <commit_hash>
```

For example, to reset to the last commit:

```
git reset --hard HEAD
```

Step 2: Force Push the Changes to the Remote

Once the reset is done locally, you can force-push the changes to overwrite the remote state:

```
git push origin <branch_name> --force
```

This will overwrite the remote branch with your local branch state.

End Tutorial # 1