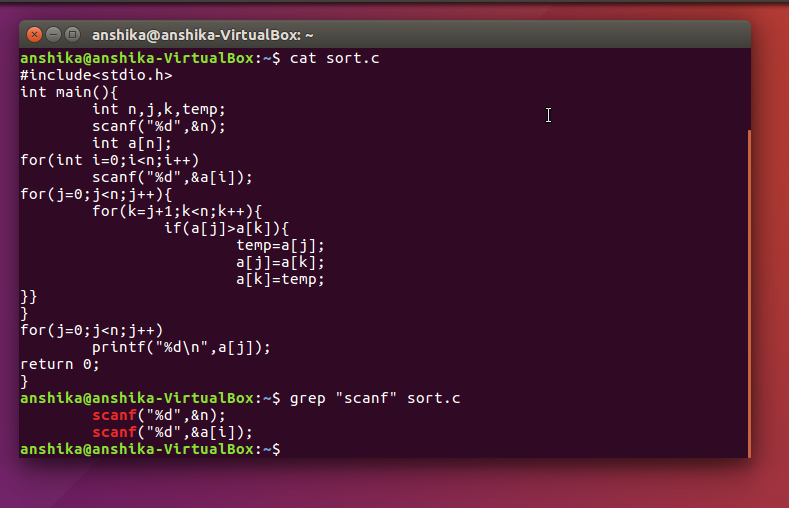
**1. Command for searching pattern line by line in any document with an example**

grep "pattern" filename.



**2. What all permissions are there in Linux? State all permissions and different way of changing permissions with example.**

There are three types of permission in Linux.

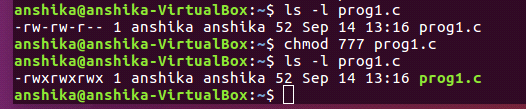
they are-

1. Read- gives permission to read a file.

2. Write- gives permission to make changes in a file.

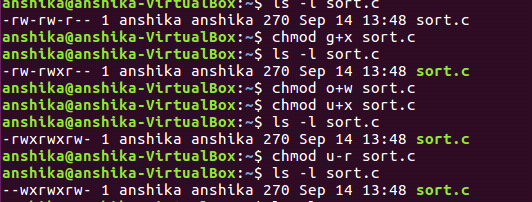
3, Execute- gives permission to execute a file.

there are two ways to change permissions.

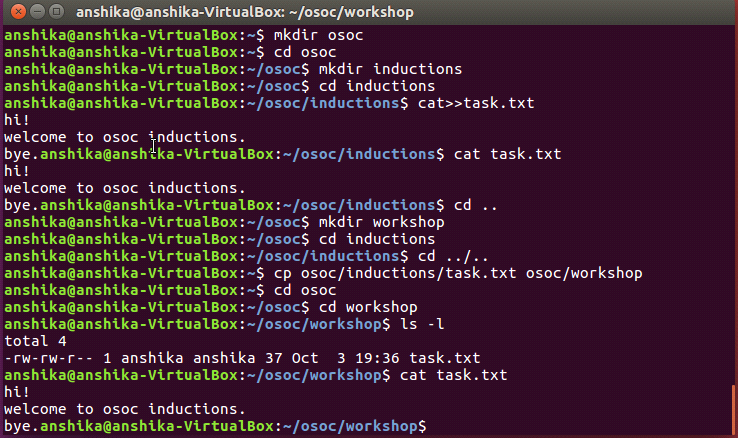
1.Absolute mode- In this mode, file permissions are represented as a three-digit octal number.

2.Symbolic mode-In the symbolic mode, you can modify permissions of a specific owner specified in command. It makes use of mathematical symbols to modify the file permissions.

command to change permissions is chmod.



**3. Create a folder *OSOC* and inside that another folder *Inductions* and now create a file *task.txt* inside *Inductions* with some content now copy this file into another folder *Workshop* in *OSOC.***



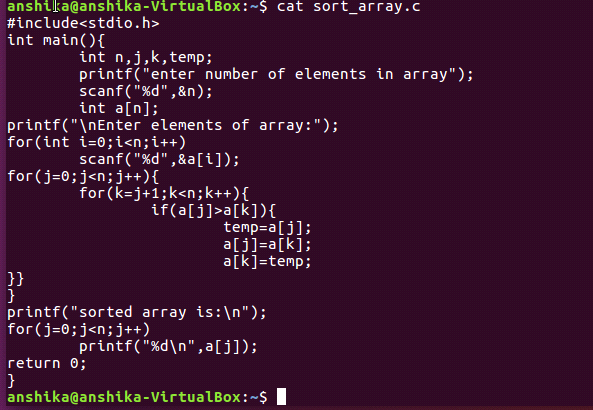
**4. I want to check whether my system is connected to any network or not, please suggest appropriate command for this.**

ping command

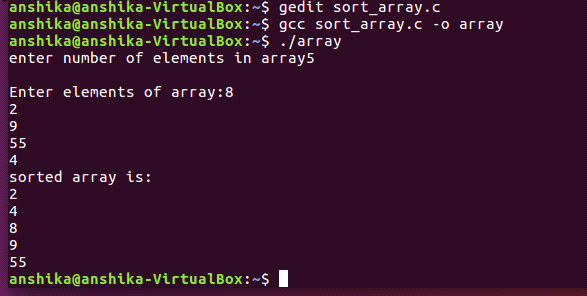
**5. I want to change password of another user how will I do that?**

To change a password on behalf of a user, first sign on "sudo" or "su" to the "root" account. Then type, ``passwd user'' (where user is the username for the password you are changing). The system will prompt you to enter a password.

**6. Using shell scripting write program for sort an Array. Input should be taken from user.**



**7. Using Shell scripting write a program to reverse a string. Input should be taken from user.**



#include<stdio.h>

#include<conio.h>

int main()

{

int i, j, k;

char str[100];

char rev[100];

printf("Enter a string:\t");

scanf("%s", str);

printf("The original string is %s\n", str);

for(i = 0; str[i] != '\0'; i++)

{

k = i;

}

for(j = 0; j <= i-1; j++)

{

rev[j] = str[k];

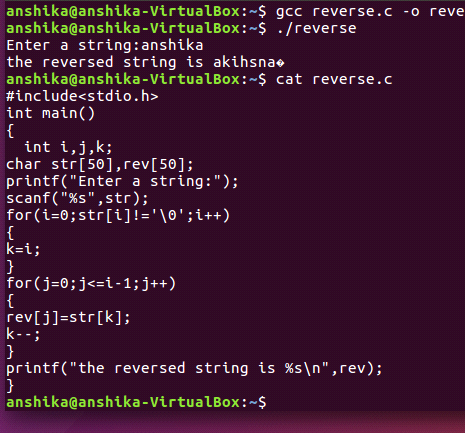
k--;

}

printf("The reverse string is %s\n", rev);

getch();

}



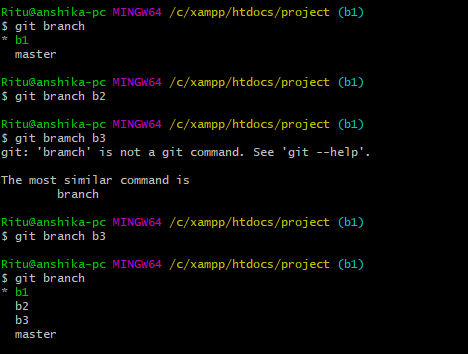
***8. What is the concept of Branching? How it is useful? Explain with working example.***

Git saves each version of your project as a snapshot of the code exactly as it was at the moment you committed it.

when more than one person is working in the project and making changes — is by using branches. A branch is essentially is a unique set of code changes with a unique name. Each repository can have one or more branches. The main branch — the one where all changes eventually get merged back into, and is called master. This is the official working version of your project, and the one you see when you visit the project repository.

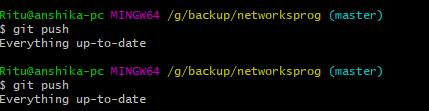
The core idea behind the Feature Branch Workflow is that all feature development should take place in a dedicated branch instead of the master branch. This encapsulation makes it easy for multiple developers to work on a particular feature without disturbing the main codebase. It also means the master branch will never contain broken code, which is a huge advantage for continuous integration environments.

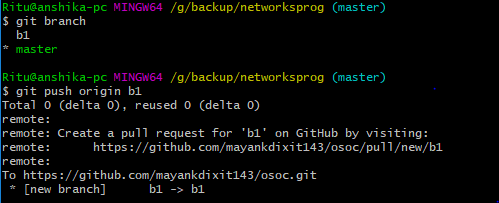
Encapsulating feature development also makes it possible to leverage pull requests, which are a way to initiate discussions around a branch. They give other developers the opportunity to sign off on a feature before it gets integrated into the official project. Or, if you get stuck in the middle of a feature, you can open a pull request asking for suggestions from your colleagues. The point is, pull requests make it incredibly easy for your team to comment on each other’s work.



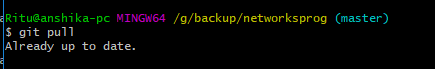
**9. What is *Push, Pull and Commit*? Explain with example.**

The git push command is used to upload local repository content to a remote repository.

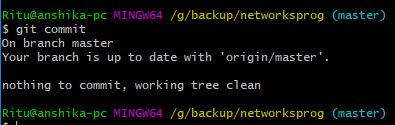




The git pull command is used to fetch and download content from a remote repository and immediately update the local repository to match that content.



committing is the process which records changes in the repository. Think of it as a snapshot of the current status of the project. Commits are done locally.



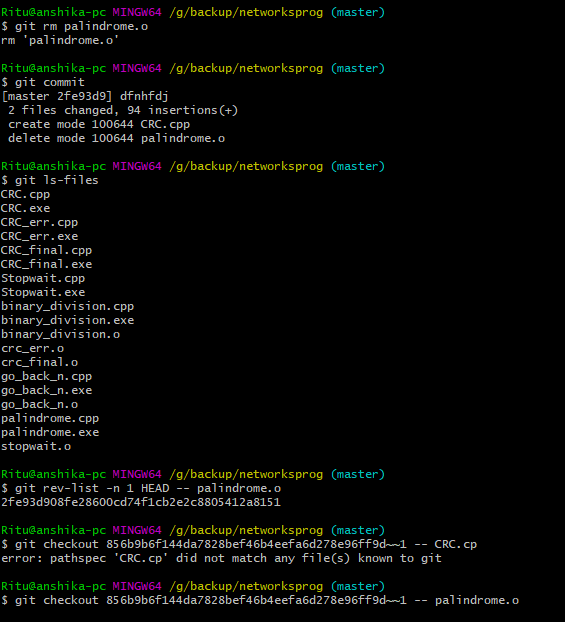
**10. Explain how we can restore a file after a commit with an Example.**

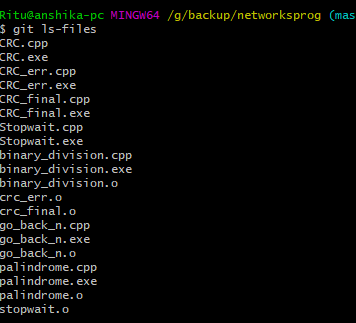
To restore a file after commit run following commands

1. git rev-list -n 1 HEAD -- <filename>

2.git checkout <commit-hash>~1 -- <file name>

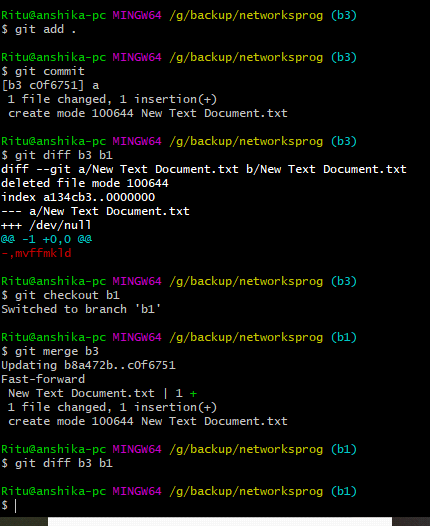
commit-hash is returned by first command

****

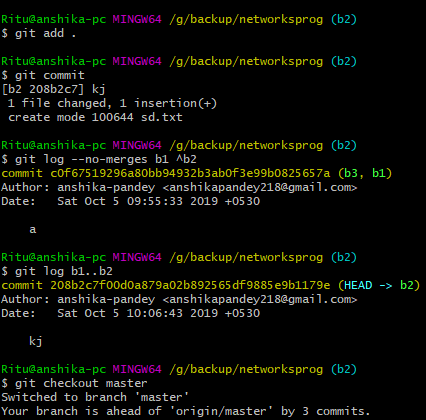
****

**11. How to preview the changes you have made before applying merging command? (write optimized command)**

git diff<source><target>

****

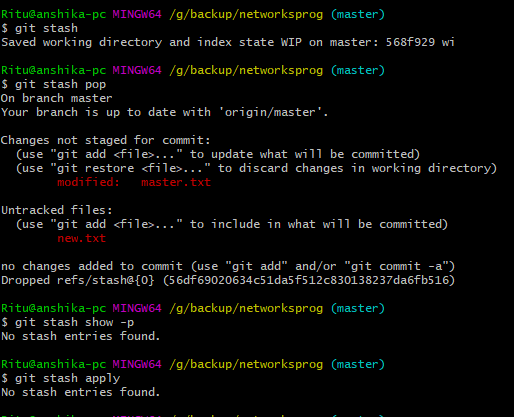
**12. How to apply any commits of current branch ahead of specified one? (write command with screenshot of command line)**



**13. what is stash stack? How to write working from top of stash stack? (write answer and show screenshot of command on command line)**

Stashing takes the dirty state of your working directory — that is, your modified tracked files and staged changes — and saves it on a stack of unfinished changes that you can reapply at any time.

When you're done with the changes, you should use git stash drop to let go of the stash-bag from the commit it was "stashed on". (And, git stash pop is just shorthand for "apply, then automatically drop



**14. How to show the commits on ex (branch Ankur) that are not on ex (branch Ruchita )? (write command with screenshot of command line)**

git log Ruchita..Ankur

