A
Report on
"Internship at Volansys "
Project
Snake Game
BLE TLV Frame Emulator

Prepared by
Swapnil Shah (17EC083)
Amartya Singh (17EC088)

Under the guidance of

Dr. Dharmendra Chauhan
Meetali Patel & Noman Cyclewala

Submitted to

Charotar University of Science & Technology for Partial Fulfillment of the Requirements for the Degree of Bachelor of Technology in Electronics & Communication EC448 - Project of 8th Semester of B.Tech

Submitted at

DEPARTMENT OF ELECTRONICS & COMMUNICATION

Faculty of Technology & Engineering,

CHARUSAT Chandubhai S. Patel Institute of Technology

At: Changa, Dist: Anand –

388421 April 2021

CHARUSAT
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY

CERTIFICATE

This is to certify that the report entitled "**VOLANSYS Internship**" is a bonafide work carried out **Swapnil Shah (17EC083)** under the guidance and supervision of **Prof. Dharmendra Chauhan** & **Meetali Patel** for the subject **Project (EC448)** of 8th Semester of Bachelor of Technology in Electronics & Communication at Faculty of Technology & Engineering (C.S.P.I.T.) – CHARUSAT, Gujarat.

To the best of my knowledge and belief, this work embodies the work of the candidate himself, has duly been completed, and fulfills the requirement of the ordinance relating to the B.Tech. Degree of the University and is up to the standard in respect of content, presentation and language for being referred to the examiner.

Under the supervision of,

Dr. Dharmendra Chauhan
Associate Professor

Department of EC Engineering,
C.S.P.I.T., CHARUSAT-Changa.

Meetali Patel
Engineer (Embedded Software & Firmware)

Volansys Technologies, Ahmedabad

Dr. Trushit K. Upadhyaya
Head of Department,
Department of Electronics & Communication
C.S.P.I.T., CHARUSAT- Changa, Gujarat.

CHARUSAT
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY

CERTIFICATE

This is to certify that the report entitled "**VOLANSYS Internship**" is a bonafide work carried out **Amartya Singh (17EC088)** under the guidance and supervision of **Prof. Dharmendra Chauhan** & **Noman Cyclewala** for the subject **Project (EC448)** of 8th Semester of Bachelor of Technology in Electronics & Communication at Faculty of Technology & Engineering (C.S.P.I.T.) – CHARUSAT, Gujarat.

To the best of my knowledge and belief, this work embodies the work of the candidate himself, has duly been completed, and fulfills the requirement of the ordinance relating to the B.Tech. Degree of the University and is up to the standard in respect of content, presentation and language for being referred to the examiner.

Under the supervision of,

Dr. Dharmendra Chauhan
Associate Professor

Department of EC Engineering,
C.S.P.I.T., CHARUSAT-Changa.

Noman Cyclewala
Engineer (Embedded Software & Firmware)

Volansys Technologies, Ahmedabad

Dr. Trushit K. Upadhyaya
Head of Department,
Department of Electronics & Communication
C.S.P.I.T., CHARUSAT- Changa, Gujarat.

Table of Contents

List of figures

List of tables

ABSTRACT

These four months of training at volansys has been a wonderful journey till now. We learnt many new things and revisited many more old things that were taught in our college curriculum. As both of us are aspiring Embedded Software & Firmware Engineers. So, the training module for us is tailored in such a way that whenever we are assigned on a project we can begin working on it while facing the least number of road blocks possible.

The training module are as follows:

- Linux Essential 1
- Git
- C Programming
- Advanced C
- Data Structures & Algorithms

These modules really helped us to bring the confidence that whenever an industrial live project is assigned to us we can at least play our role in that well.

We would also like to thank our college for providing such opportunities in the last semester that make students ready for industries.

ACKNOWLEDGEMENT

We would like to express my sincere thanks to our internal guide **Prof. Dharmendra Chauhan** for his valuable guidance, enthusiastic attitude and encouragement throughout the period of Project and Thesis work. His guidance, suggestion and very constructive appreciation have contributed immensely to the evolution of the project.

We would also like to thank our external guide **Meetali Patel & Noman Cyclewala** for helping us during our project work. It was due to their constant guidance and supervision that we were able to understand all the concepts thoroughly and complete our project work.

We would also like to express our sincere gratitude and sincere thanks to **Dr. Trushit Upadhyaya** HoD, Department of Electronics and Communication Engineering, Charotar University of Science and Technology, Changa for providing us the opportunity to undertake such challenging and intellectual work.

Swapnil Shah (17EC083)
Amartya Singh (17EC088)

Department of EC Engineering,

CSPIT, CHARUSAT

INTRODUCTION

- To be a good Embedded Firmware Engineer one needs to have good knowledge of c programming language, Linux operating system, GitLab, Data Structures etc. In my training I have gone through all these very well-prepared training modules by Volansys and also completed all the exercises in each module.

- First module of this report is on Linux basics which covers all the basic command, user & administrations commands, shell scripts and vi editor all these are very essential to work efficiently with Linux operating system.

- Second module of this report is on GitLab which covers git structure and how to use GitLab to collaborate with team member to work on any project which is very basic need in industry and it also covers all the basic GitLab command along with GitLab continuous integration which is very helpful to make development process very fast.

- Third module of this report is on c programming language which is the base of the embedded firmware engineering and covers all the basic concepts of the c language with good examples to clear all the concepts.

- Fourth module is on advanced c which will take you depth in c and cover many interesting aspects of c programming like memory organization, makefile to compile your code automatically when you change any file and tools like valgrind to make your program more optimized, faster and correct.

- Fifth module is on Data Structures in which we learned to create a single, double, circular linked list. We also learnt about different types of operations and modes in which link lists can be operated. Finally, we learnt about sorting algorithms, binary trees and binary search trees.

- The Last module is on Mega Exercise in which whatever we have learned so far, have implemented in this Exercise.

Chapter 1: Linux Training

1.1 Understanding the shell

- **Shell**

  Shell is basically a command interpreter. It provides an interface between the user and operating system. It accepts input commands from the user and executes programs according to input and when execution of the program completes it shows output to the user.

- **Types of Shells**

  - c shell: denoted as csh
  - bourne shell: denoted as sh
  - korn shell: ksh
  - bourne again shell:bash
  - z shell: zsh

- **Finding current shell in use**

  -echo $SHELLS

1.2 Environment variables

Environment (GLOBAL) variables are variables that are available system-wide and are inherited by all spawned child processes and shells. Environment variables or ENVs basically define behaviour of the environment. They can affect the ongoing processes.

- **Example**

USER

The current logged in user.

HOME

The home directory of the current user.

EDITOR

The default file editor to be used.

SHELL

The path of the current user's shell, such as bash or zsh.

PATH

A list of directories to be searched when executing commands. When you run a command the system will search those directories in this order and use the first found executable.

- **env:** it will print all environment variables.
- **printenv:** it will print all environment variables.
- **set:** it will print all environments as well as shell variables.

1.3 Aliases

An alias is a shortcut command to a longer command. Users may type the alias name to run the longer command with less typing.A new alias is defined by assigning a string with the command to a name.

Alias are often set in the ~/.bashrc file.

- **Creating an alias :** Syntax:

  alias name="value"

- **Creating an Unalias**

  Removing an existing alias is known as

  unaliasing. Syntax: unalias [alias name]

  alias -p: it will print all the defined alias.

- **How to edit the shell rc file to add one alias to any command.**

  vim .bashrc : to open .bashrc file

  enter i: to go into insert mode

  alias CD=”cd /person”: it will create an alias with name CD.

  Esc: to return back into command mode.

  :wq : to save and quit.

  source .bashrc : to reload .bashrc file.

1.4 Job/Process control commands

ps =>It will print the process running in the current shell.

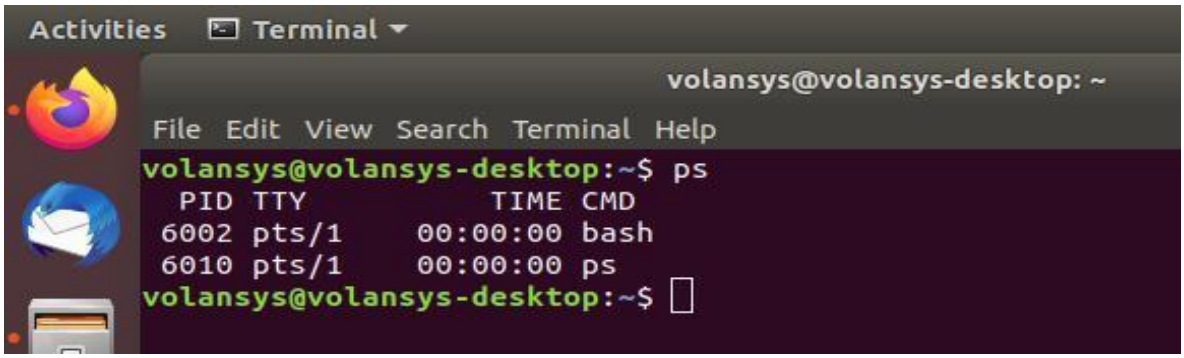


Fig.1.1 output of ps command

ps -e =>it will print every running process

ps -ef =>it will print every running process with full formatting.

free =>It will display amount of free & used memory in the system

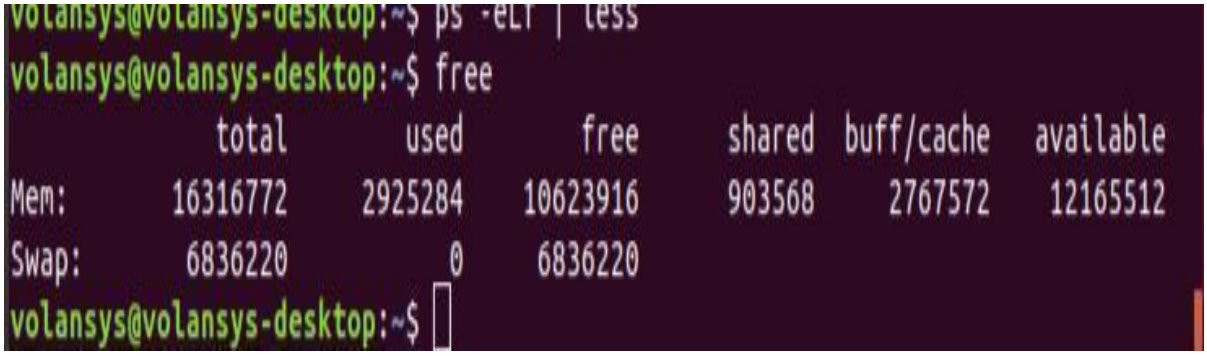


Fig.1.2 output of free command

free -b =>It will display memory in bytes **free -l** =>it will display high and low memory statistics

- **Running a process in background**

  If we want to run any process in background then we need to use "&" at the end of the command or we can use (cltr+z) to sleep the process after that run bg command to put that process in background. We can use the jobs command to show processes in the background.

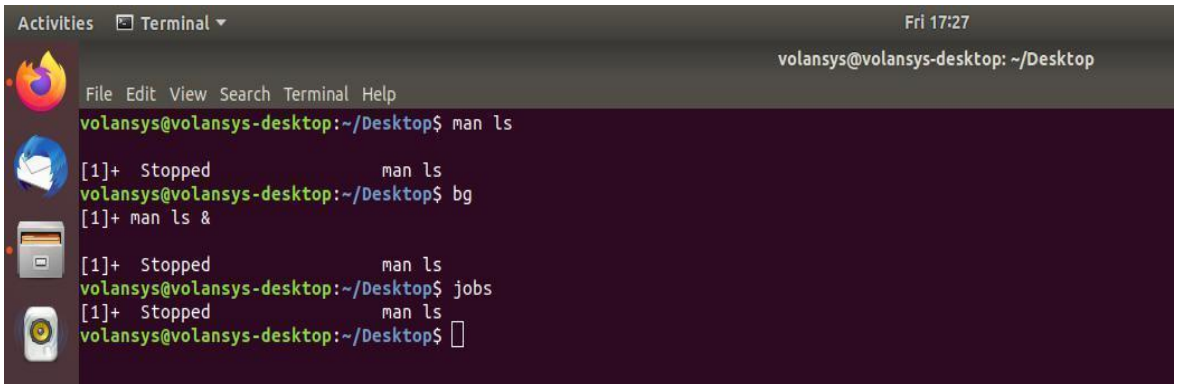


Fig.1.3 output of sending process in background

Moving background process to foreground

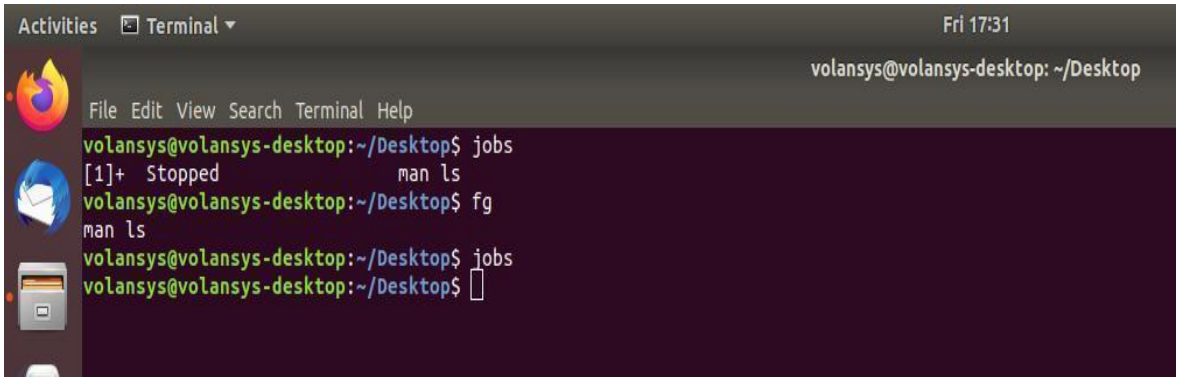


Fig.1.4 output of moving background process to foreground

kill pid => it will kill process with given pid

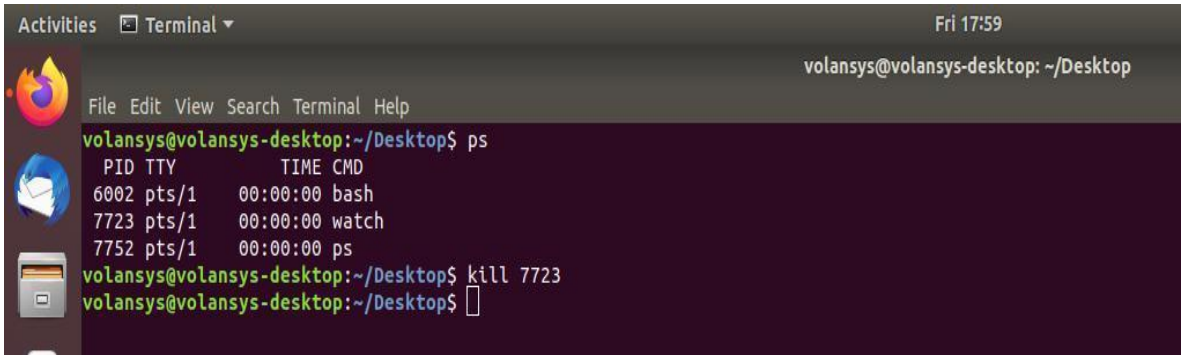


Fig.1.5 output of kill command

kill -9 pid =>it will forcefully kill process with given pid

pkill process-name => it will kill process by its name

1.5 User management control command

who =>it will display who is logged in

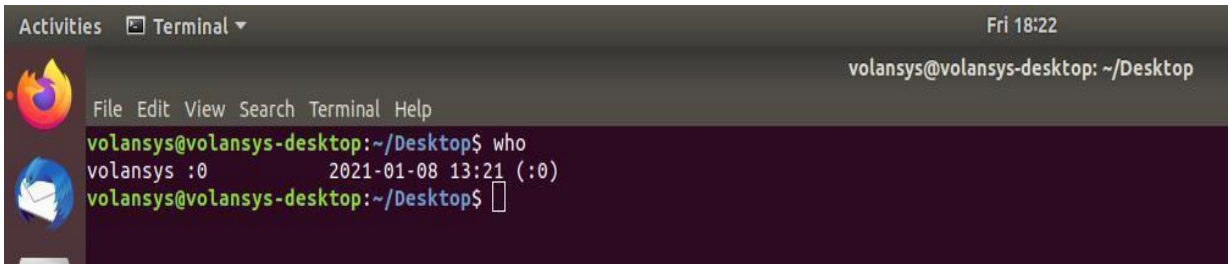


Fig.1.6 output of who command

who -b =>it will print time of last system boot **who -q** =>it will print all login names & number of user logged in

finger =>it will display the information about the user

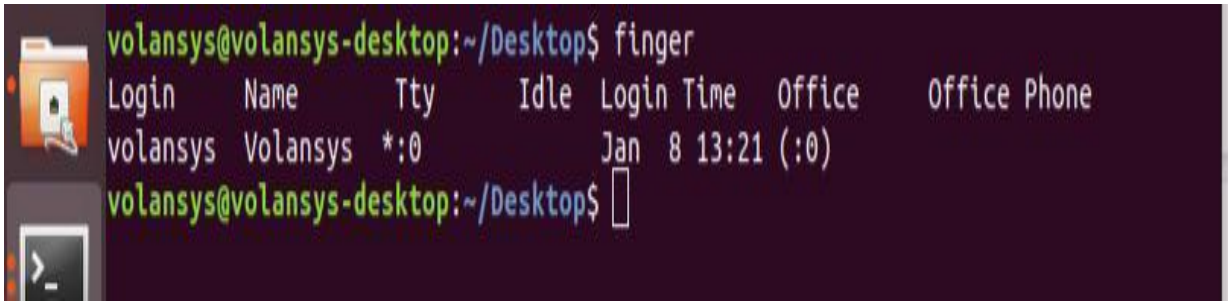


Fig.1.7 output of finger command

- **Super user**

Super user has all the privileges. Super user has all read,write and execute permissions,creating or installing files or software,deleting and creating users**.**

- **Switching to Super User**

sudo su =>it will allow current user to switch to the super user

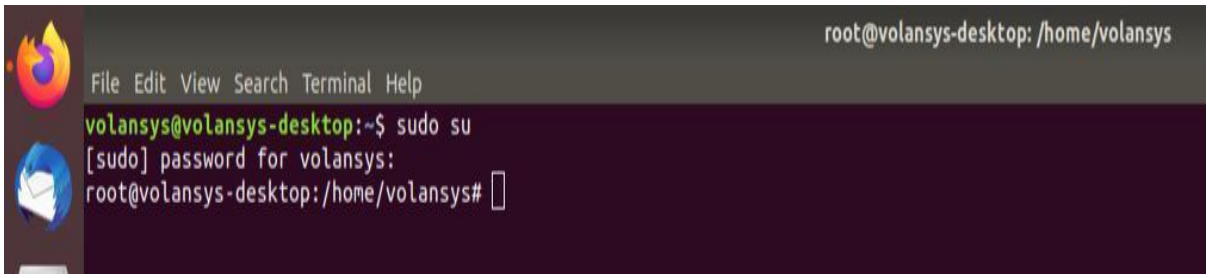


Fig.1.8 output of su command

- **Create a user, delete a user**

Useradd person => it will create user named person

passwd person =>it will allow user named person to change or set the password

```
volansys@volansys-desktop:~$ sudo -i
root@volansys-desktop:~# useradd person
root@volansys-desktop:~# passwd person
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

```
volansys@volansys-desktop:~$ cat /etc/passwd | grep person
person:x:1001:1001::/home/person:/bin/sh
volansys@volansys-desktop:~$
```

Fig.1.9 output of useradd & passwd command

userdel person => it will delete the user named person

```
root@volansys-desktop:~# userdel person
root@volansys-desktop:~#
root@volansys-desktop:~#
root@volansys-desktop:~#
root@volansys-desktop:~#
root@volansys-desktop:~# cat /etc/passwd | grep person
```

Fig.1.10 output of userdel command

1.6 Network administration command

- **ifconfig**

  ifconfig command is used to show current network configuration and setting up ip address ,netmask or broadcast address , creating aliases for network interface, setting MAC address and enabling or disable network address.

Ifconfig -a =>it will display all current network configuration

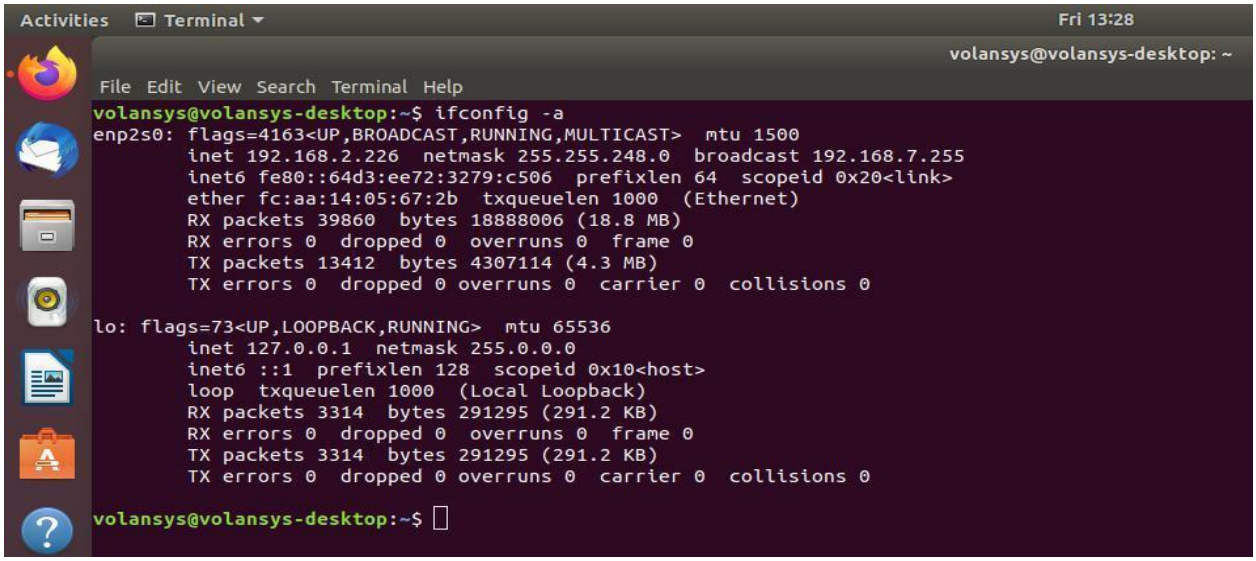


Fig.1.11 output of ifconfig command

Ifconfig lo =>it will display specific network configuration

Ifconfig -s =>it will display short list

- **Ifconfig enp2s0 up =>** If address is assigned to this interface then it will activate this interface(driver of this interface)
- **Ifconfig enp2s0 down =>**It will deactivate driver for this interface

- **Apropos**

  Apropos command helps the user in finding the exact command by giving some known few keywords related to the command. It searches through the man page of Linux with the keyword provided.

- **Configuration of the network interface for DHCP.**

  ==>

  sudo vim /etc/network/interfaces

  In this file copy following & comment the previous code.

  auto <interface name>

  iface <interface name> inet dhcp

  Then disable and then enable networks using.

  $ sudo ip a flush enp2s0

$ sudo systemctl restart networking.service

- **Configuration of the network interface for static ip address. ==>**

  Sudo vim /etc/network/interfaces

  In this file copy following & comment the previous code.

  auto <interface name>

  iface <interface name> inet static

  address 192.168.1.1 -keep according to your need

  netmask 255.255.255.0 -keep according to your need

  gateway 192.168.0.1 -keep according to your need

  dns-nameservers 4.4.4.4 -keep according to your need

  Then disable and then enable networks using.

  $ sudo ip a flush enp2s0

  $ sudo systemctl restart networking.service

1.7 System administration command

Mount

==>

mount /dev/sdb1 ./swapnil =>it will mount device sdb1(usb) on dir swapnil.

df =>it will show space available in each filesystem.

Umount

==>

umount /home/volansys/swapnil =>it will unmount the device(usb) which is mounted on the dir swapnil.

```
volansys@volansys-desktop:~$ sudo mount -l -t ext4
/dev/sda3 on / type ext4 (rw,relatime,errors=remount-ro,data=ordered)
volansys@volansys-desktop:~$ sudo mount -l -t fuseblk
volansys@volansys-desktop:~$ sudo mount /dev/sda4 /media/swapnil
volansys@volansys-desktop:~$ sudo mount /dev/sda5 /media/swapnil
mount: /media/swapnil: unknown filesystem type 'swap'.
volansys@volansys-desktop:~$ sudo mount /dev/sda6 /media/swapnil
volansys@volansys-desktop:~$ sudo mount -l -t ext4
/dev/sda3 on / type ext4 (rw,relatime,errors=remount-ro,data=ordered)
/dev/sda4 on /media/swapnil type ext4 (rw,relatime,data=ordered)
/dev/sda6 on /media/swapnil type ext4 (rw,relatime,data=ordered)
volansys@volansys-desktop:~$ sudo mount -l -t fuseblk
volansys@volansys-desktop:~$ sudo unmount /dev/sda6
sudo: unmount: command not found
volansys@volansys-desktop:~$ sudo umount /dev/sda6
volansys@volansys-desktop:~$ sudo umount /dev/sda4
volansys@volansys-desktop:~$ sudo mount -l -t ext4
/dev/sda3 on / type ext4 (rw,relatime,errors=remount-ro,data=ordered)
volansys@volansys-desktop:~$
```

Fig.1.12 output of mount/unmount command

Mkfs

== > sudo mkfs /dev/sdb1 =>it will create filesystem on device /dev/sdb1 and we Have not provide type of file system so it will take ext4 type by default and the content of dev/sdb1 will be lost.

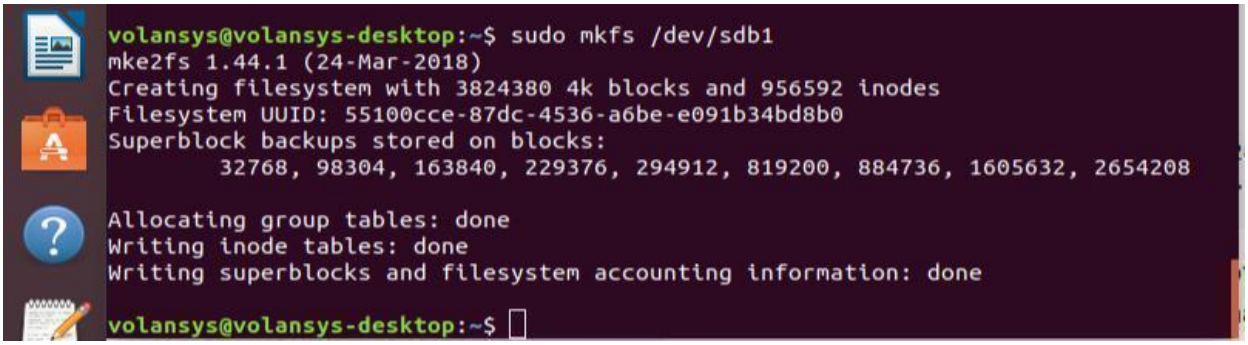


Fig.1.13 output of mkfs command

Shutdown

==>This command can be used to shutdown ,hault ,reboot the system.

-H, Halt the machine.

-P, Power-off the machine (the default).

-r, Reboot the machine.

-k,Do not halt, power-off, reboot, just write a wall message.

--no-wall, Do not send wall messages before halt, power-off, reboot.

-c, Cancel a pending shutdown.

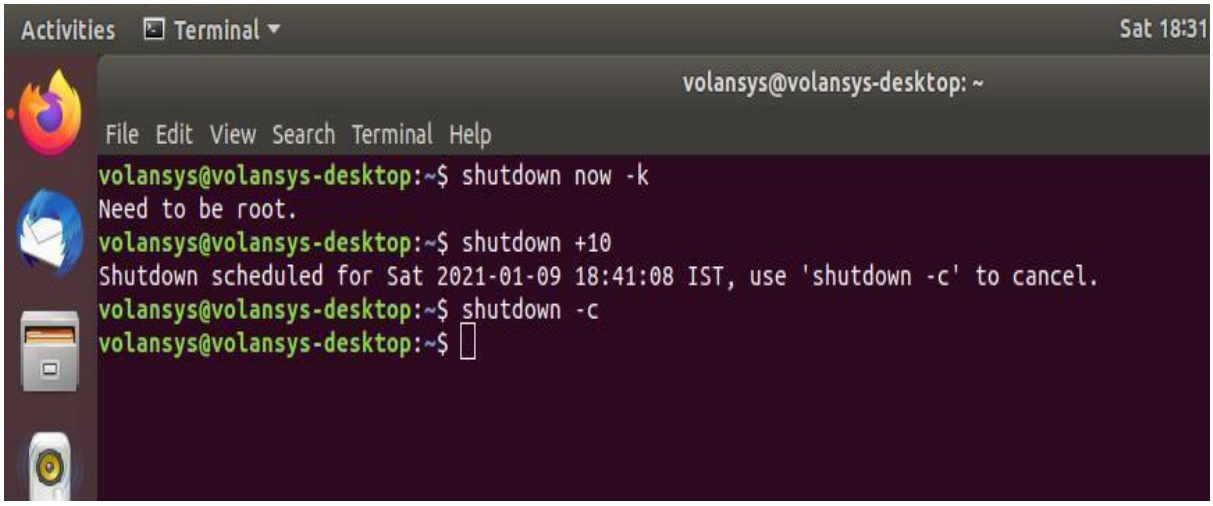


Fig.1.14 output of shutdown command

passwd==>Change password of a user

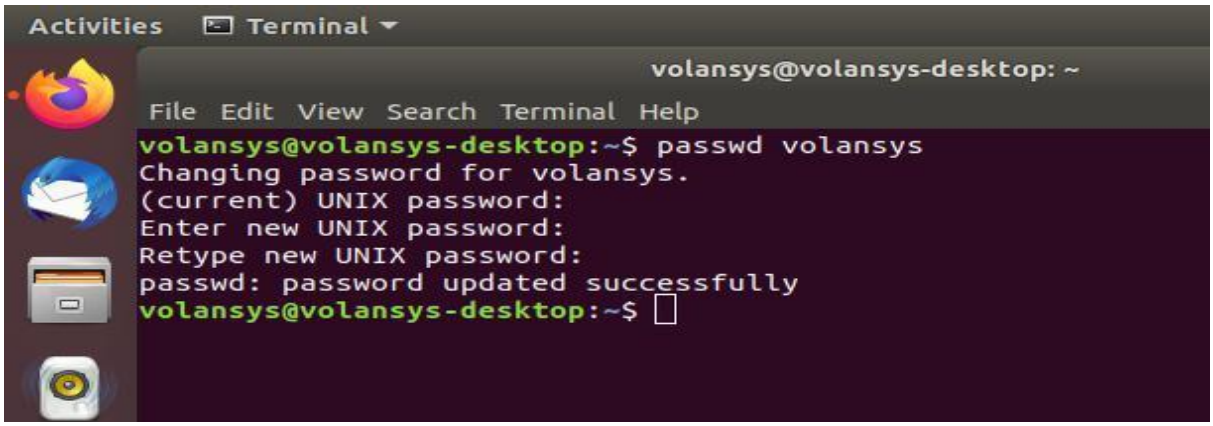


Fig.1.15 output of passwd command

Fsck

==>fsck command is used to check and repair the file system.

sudo fsck -N /dev/sdb1 =>it will not take any action it will simply print what will be done.

sudo fsck /dev/sdb1 =>it will perform a check operation on /dev/sdb1

sudo fsck -y /dev/sdb1 =>it will perform repair operation on /dev/sdb1

sudo fsck -p /dev/sdb1 =>it will perform a automatic operation on device without intervention of the user.

sudo -AR =>it will check all filesystems listed under /etc/fstab & skip root

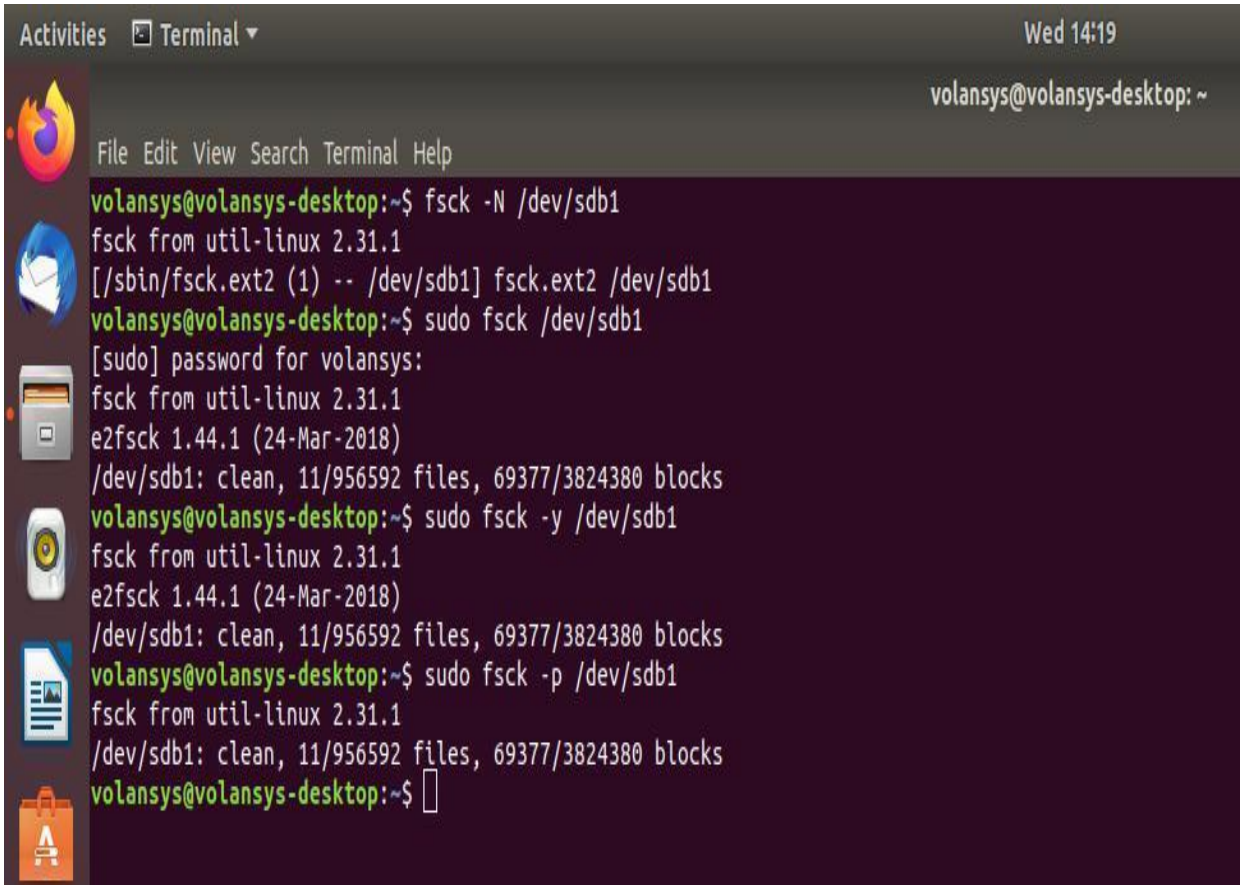
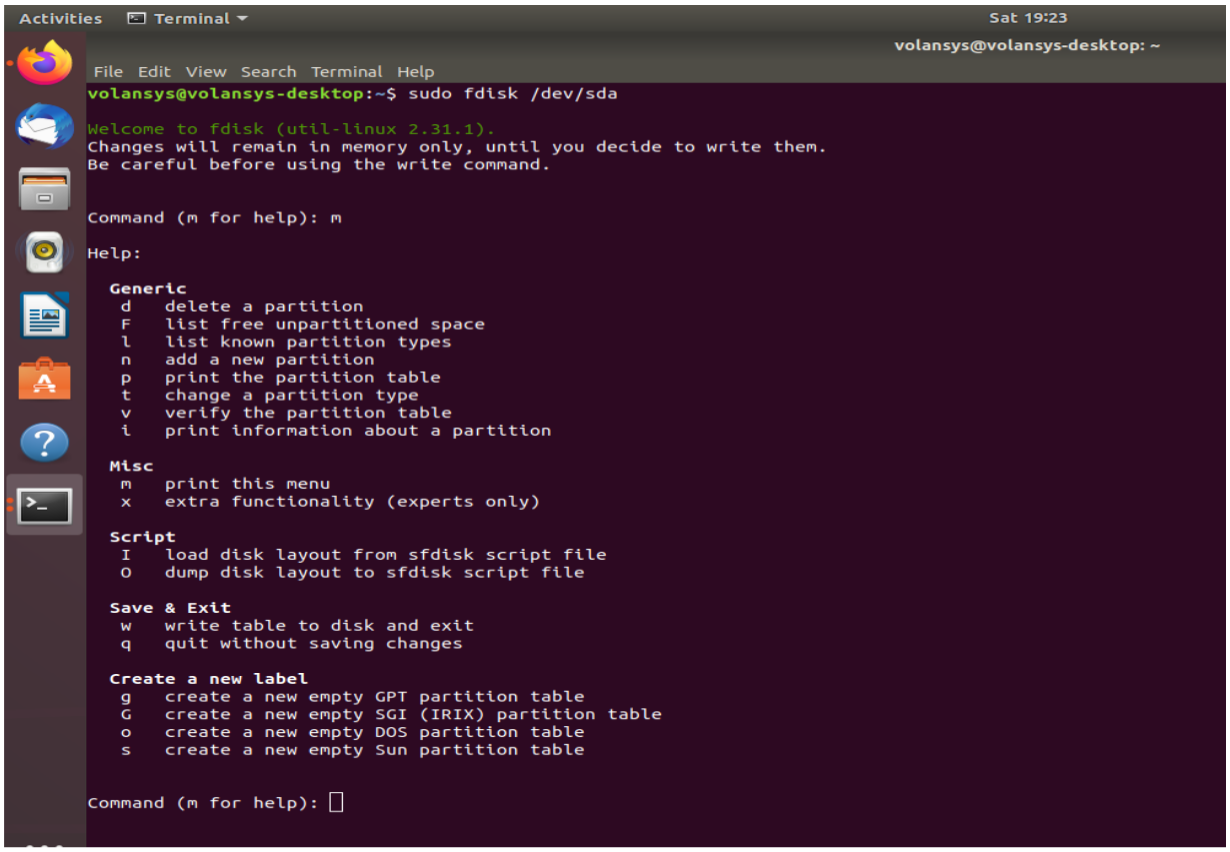


Fig.1.16 output of fsck command

Fdisk

==>



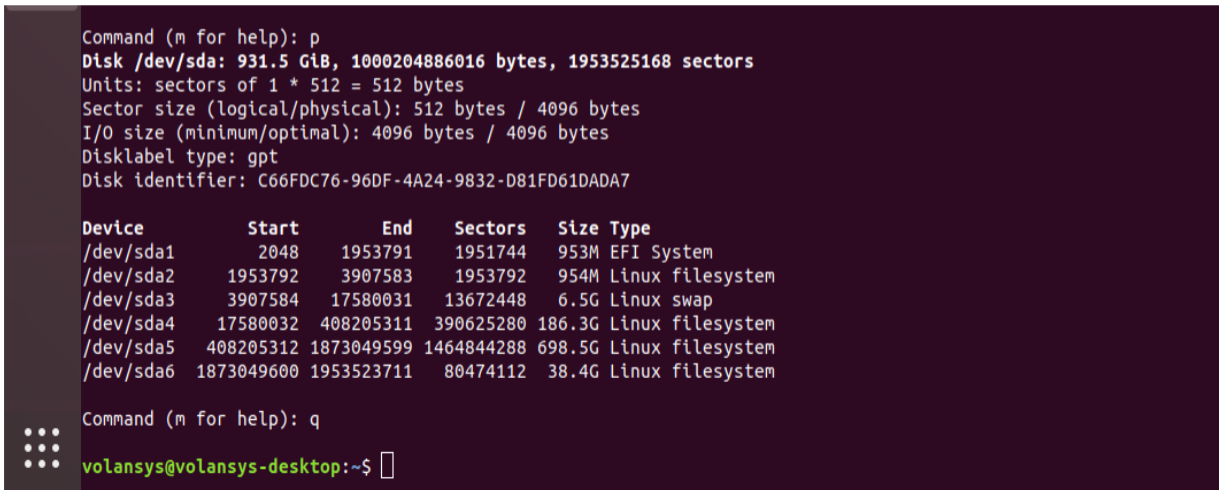


Fig.1.17 output of fdisk command

1.8 apt-get installation

==>

apt-get install iperf=>it will install package iperf. apt-get remove iperf=>it will remove package iperf not its configurations. apt-get purge iperf=>it will remove package iperf and its configuration both. apt-get update=>it will not install the package but simply update the package

which are already installed.

apt-get upgrade =>it will install the latest version of all packages listed under

/etc/apt/sources.list

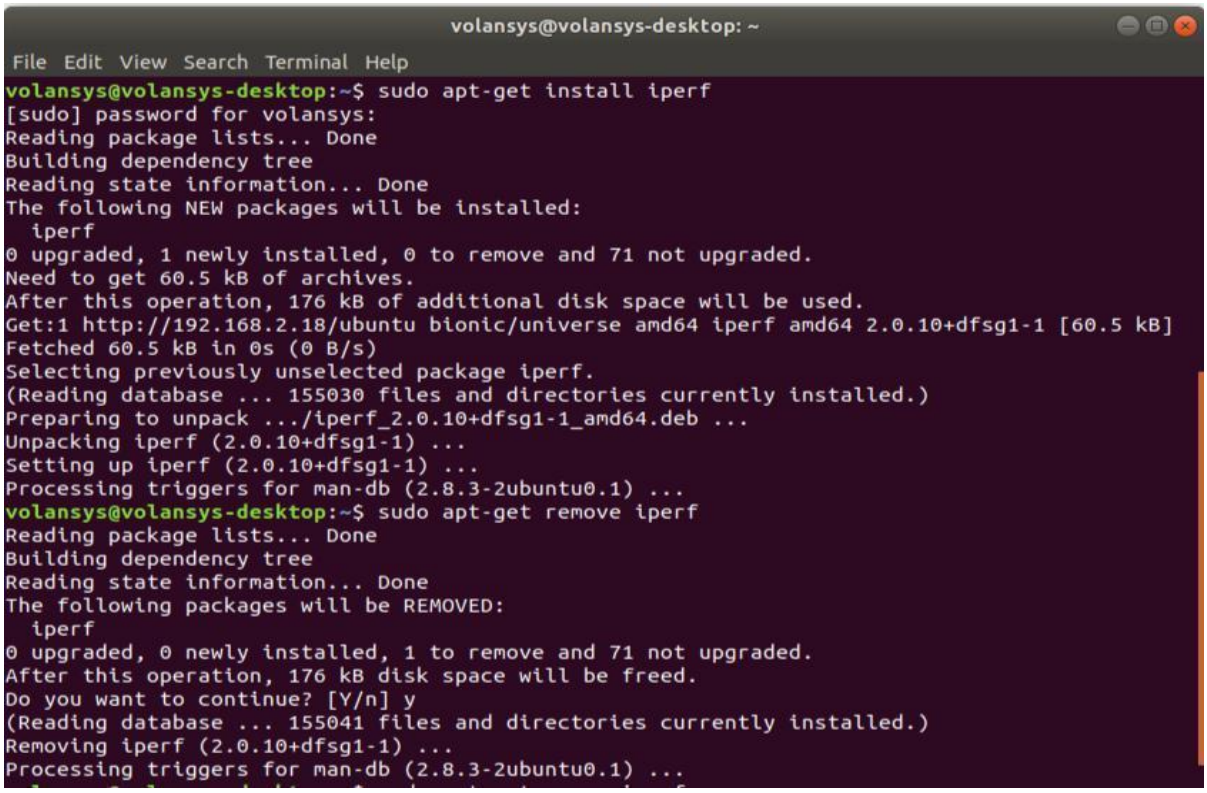


Fig.1.18 output of apt-get install & apt-get remove command

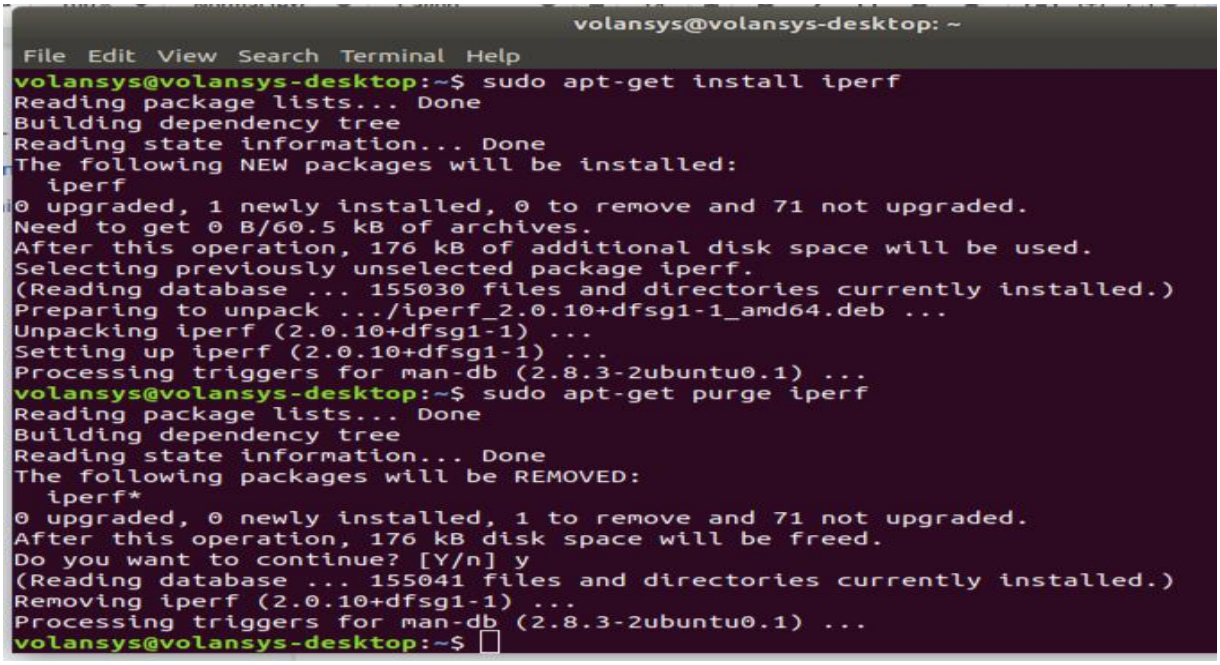


Fig.1.19 output of apt-get purge command

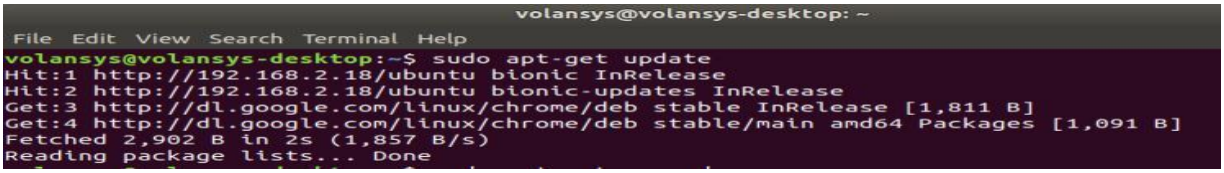


Fig.1.20 output of apt-get update command

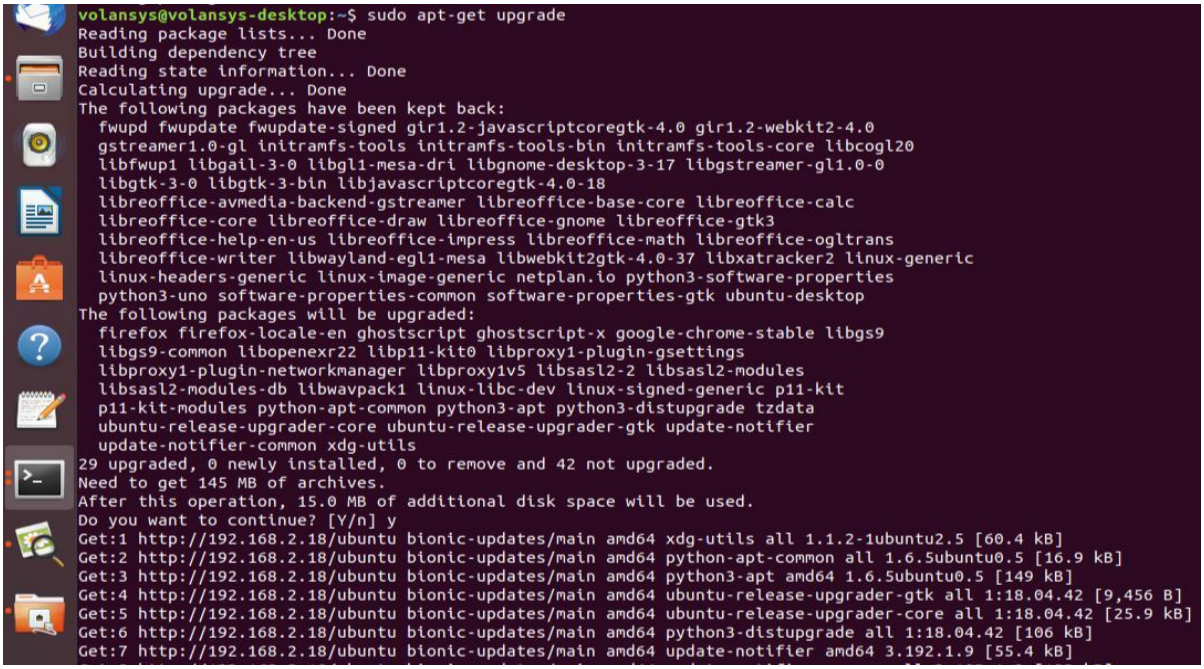


Fig.1.21 output of apt-get upgrade command

1.9 Vi editor

Command	Command Function
i	Insert at cursor (goes into insert mode)
a	Write after cursor (goes into insert mode)
A	Write at the end of line (goes into insert mode)
ESC	Terminate insert mode
u	Undo last change
U	Undo all changes to the entire line
o	Open a new line (goes into insert mode)
dd	Delete line
3dd	Delete 3 lines.
D	Delete contents of line after the cursor
C	Delete contents of a line after the cursor and insert new text.
dw	Delete word
4dw	Delete 4 words
cw	Change word
x	Delete character at the cursor
r	Replace character
R	Overwrite characters from cursor onward
s	Substitute one character under cursor continue to insert
S	Substitute entire line and begin to insert at the beginning of the line
~	Change case of individual character

Table 1.1 vi editor commands

ctags

==>

Ctags is a very useful tool to navigate any source code of the programming language. Identifiers, methods, classes, etc. from the source code are parsed by using ctags and saved in a tag file. Each tag is stored in each line. This tool helps the user to search any method or function block to find out how it works.

First move to the directory to which we want to make tag files.

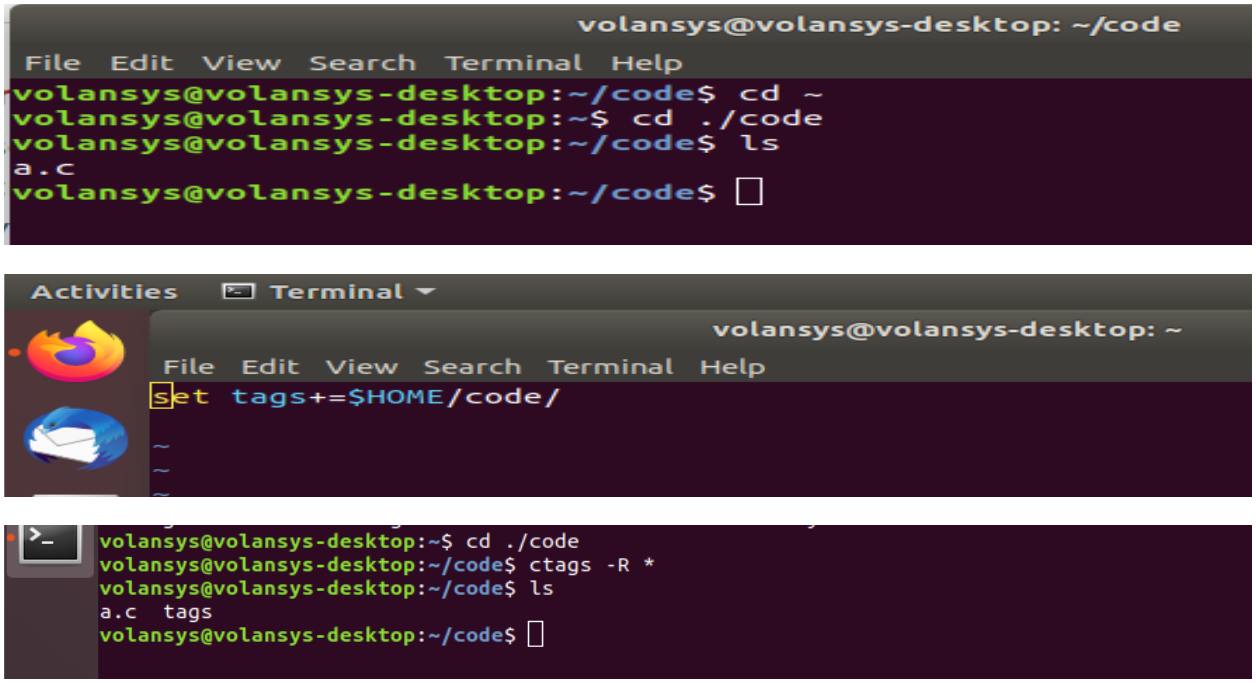


Fig.1.22 creating a tag file using ctag

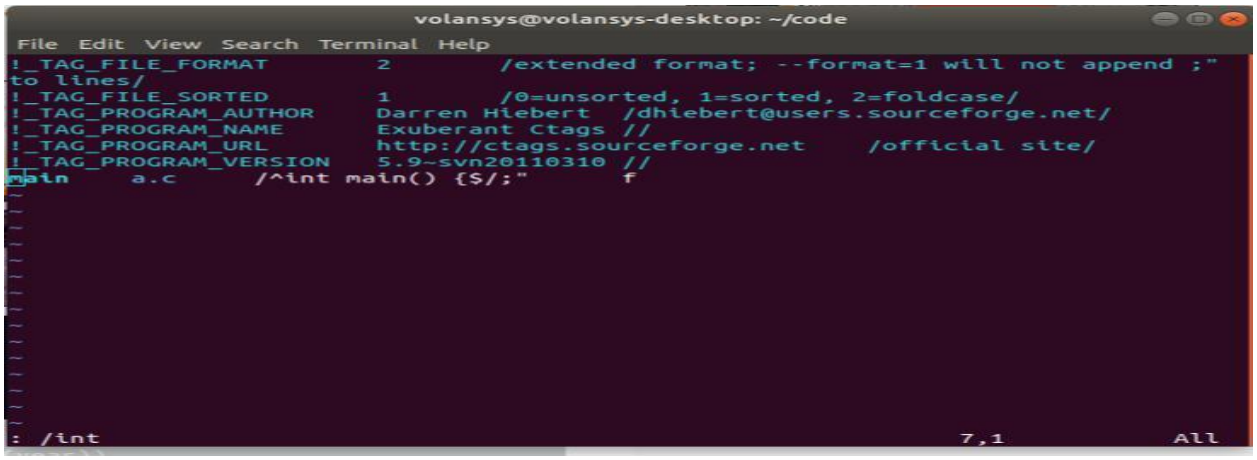


Fig.1.23 searching into tagfile using pattern

1.10 Redirection

1.10.1 '>' and '>>'

==>

'>' it will overwrite the previous content of the file and '>>' it will append after the previous content of the file.

```
volansys@volansys-Desktop:~$ ls
18.04.sh  demo2.c  Desktop    examples.desktop  Public     Training
a.out     demo.c   Documents  Music             snap       Videos
demo1.c   demo.s   Downloads  Pictures          Templates
volansys@volansys-Desktop:~$ ls -l | grep Videos >> text
volansys@volansys-Desktop:~$ cat text
drwxr-xr-x 2 volansys volansys 4096 Feb 18 17:33 Videos
volansys@volansys-Desktop:~$ ls -l | grep Training >> text
volansys@volansys-Desktop:~$ cat text
drwxr-xr-x 2 volansys volansys 4096 Feb 18 17:33 Videos
drwxr-xr-x 9 volansys volansys 4096 Apr 19 16:05 Training
```

Fig.1.24 output of redirection using '>>'

```
volansys@volansys-Desktop:~$ ls -l | grep Training > text
volansys@volansys-Desktop:~$ cat text
drwxr-xr-x 9 volansys volansys 4096 Apr 19 16:05 Training
volansys@volansys-Desktop:~$
```

Fig.1.25 output of redirection using '>'

1.10.2 '<'

==>

'<' it is used as input redirection.

```
volansys@volansys-Desktop:~$ ls -l > test
volansys@volansys-Desktop:~$ grep Videos < test
drwxr-xr-x 2 volansys volansys 4096 Feb 18 17:33 Videos
volansys@volansys-Desktop:~$
```

Fig.1.26 output of redirection using '<'

1.10.3 tee

==>

```
volansys@volansys-Desktop:~$ ls -l | grep Training > text
volansys@volansys-Desktop:~$ echo $SHELL | tee -a text
/bin/bash
volansys@volansys-Desktop:~$ cat text
drwxr-xr-x 9 volansys volansys 4096 Apr 19 16:05 Training
/bin/bash
volansys@volansys-Desktop:~$
```

Fig.1.27 output of redirection using tee

1.11 Pipes

1.11.1 “|” symbol

==>

```
volansys@volansys-desktop:~/swapnil$ cat b.txt | head -4
Tamil Nadu
Gujarat
Andhra Pradesh
Bihar
```

Fig.1.28 output of ‘|’ symbol

1.11.2 more and less commands

==>

more and less command is used to display the content of the file & navigate through a file.

enter key : navigate line by line.

spacebar key: navigate page by page.

q : quite the command.

more

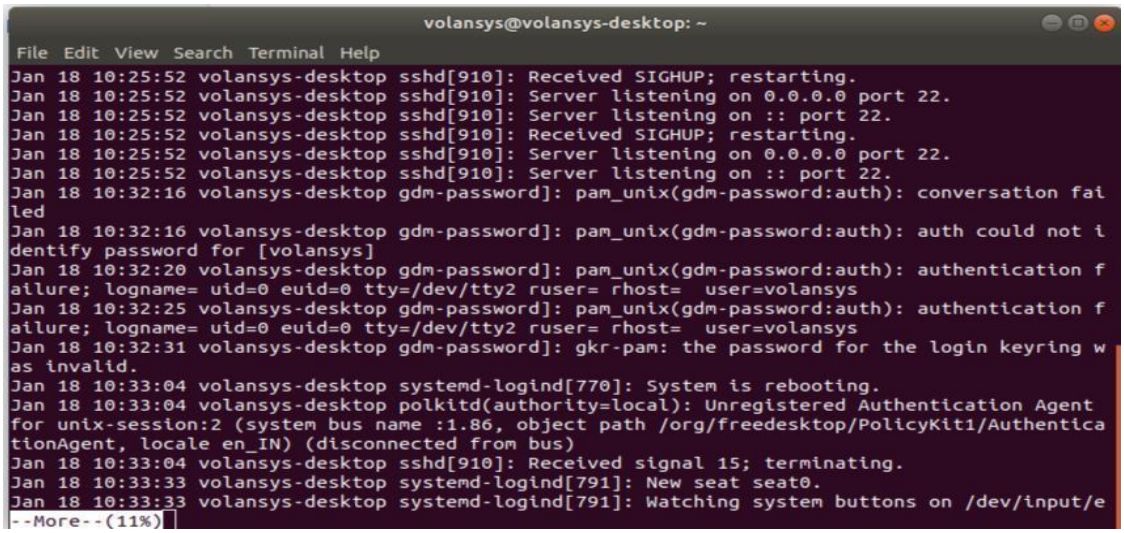


Fig.1.29 output of more command

-10 =>it will set the 10 number of lines in one page.

+10 =>it will display the output from line number 10.

less

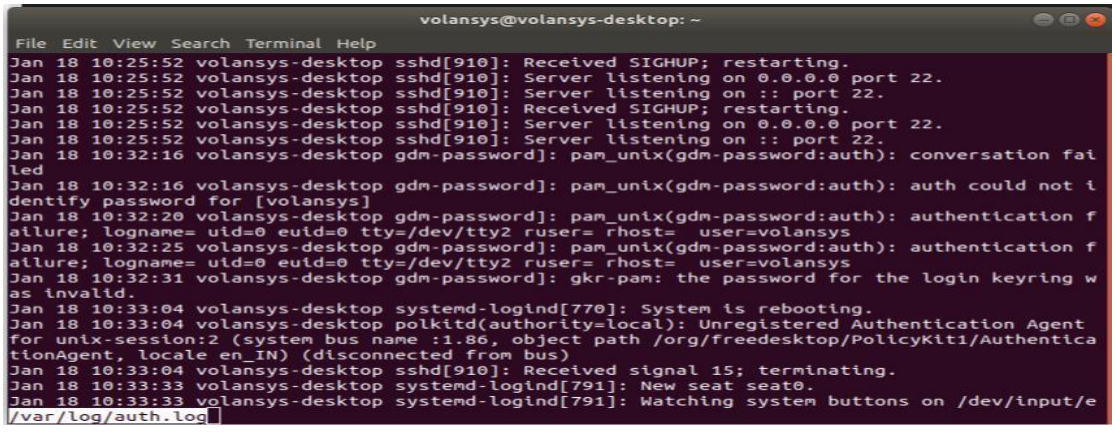


Fig.1.30 output of less command

+10 =>it will display the output from line number 10.

-N =>it will display the output along with line number.

/volansys =>it will highlight the string volansys.

1.11.3 head and tail command

head file ==> shows first 10 line of the file

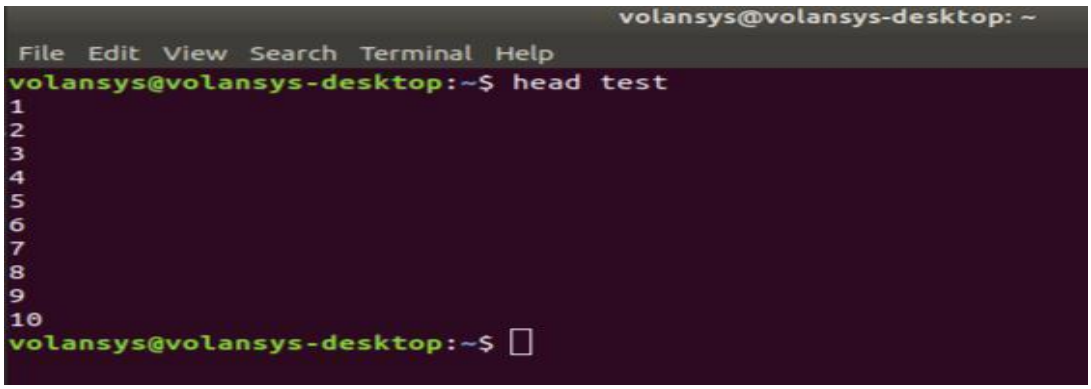


Fig.1.31 output of head

command head -n 5 file ==> shows first 5 line of the file

head -n -5 file ==> shows all line of the file except last 5 lines

head -n -5 file1 file2 ==> shows first 5 lines of file1 and file2

tail file ==> shows the last 10 lines of the file.

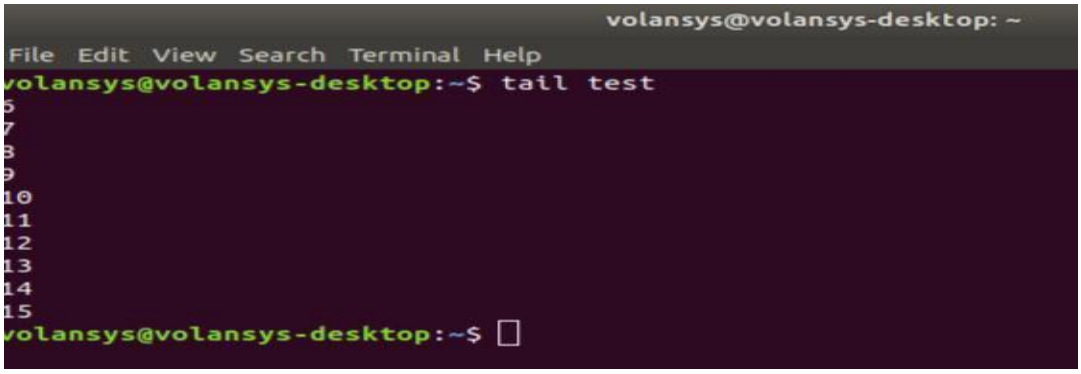


Fig.1.32 output of tail command

tail -n 5 file ==> shows the last 5 lines of the file.

tail -n +5 file ==> shows all the lines of the file except the first 5 lines.

tail -n 5 file1 file2 ==> shows the last 5 lines of file1 and file2.

1.12 Services

Service --status-all ⇒ shows status of all the services.

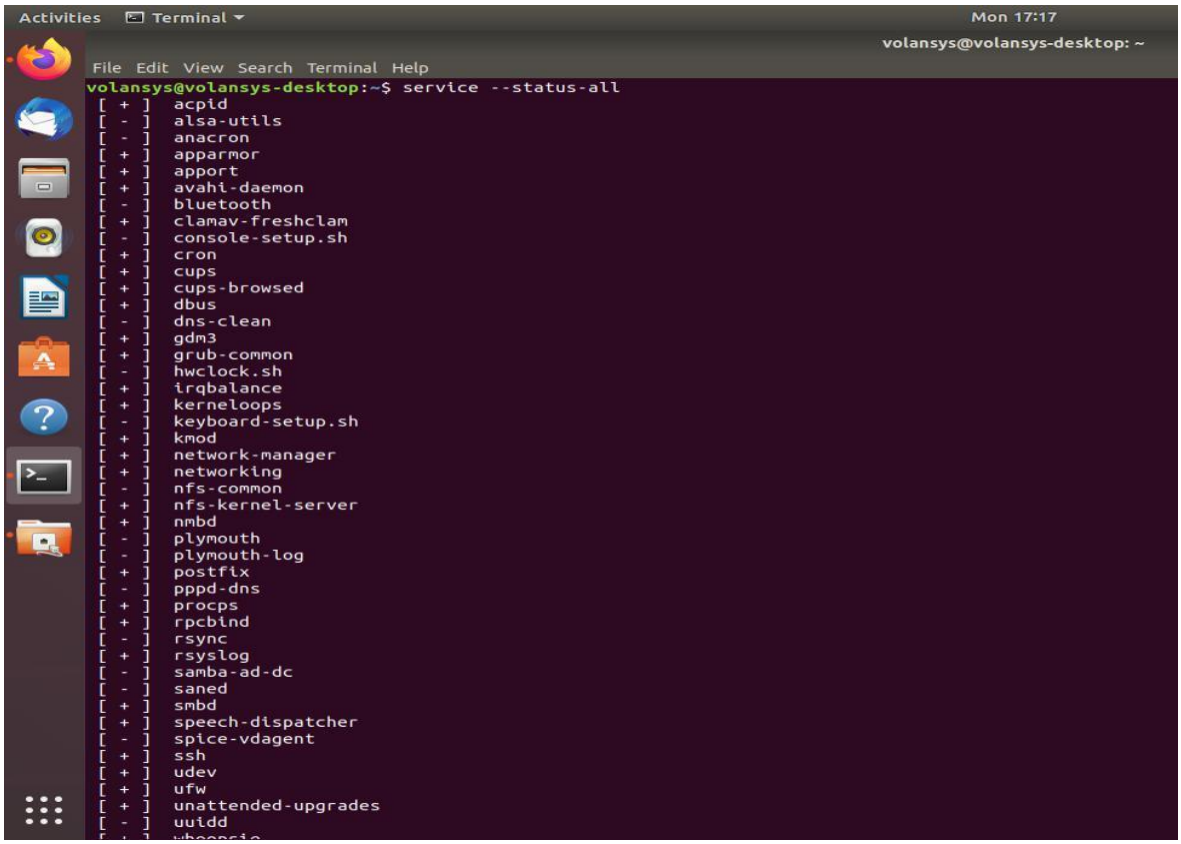


Fig.1.33 output of service command

Service --status-all | grep nfs ⇒ shows status of the nfs service.

Service nfs-kernel-server stop ⇒ stops the nfs service.

Service nfs-kernel-server start ⇒ starts the nfs service.

Service nfs-kernel-server restart ⇒ restarts the nfs service.

SSH command

ssh ip1 ==> to login to the system with ip address ip1

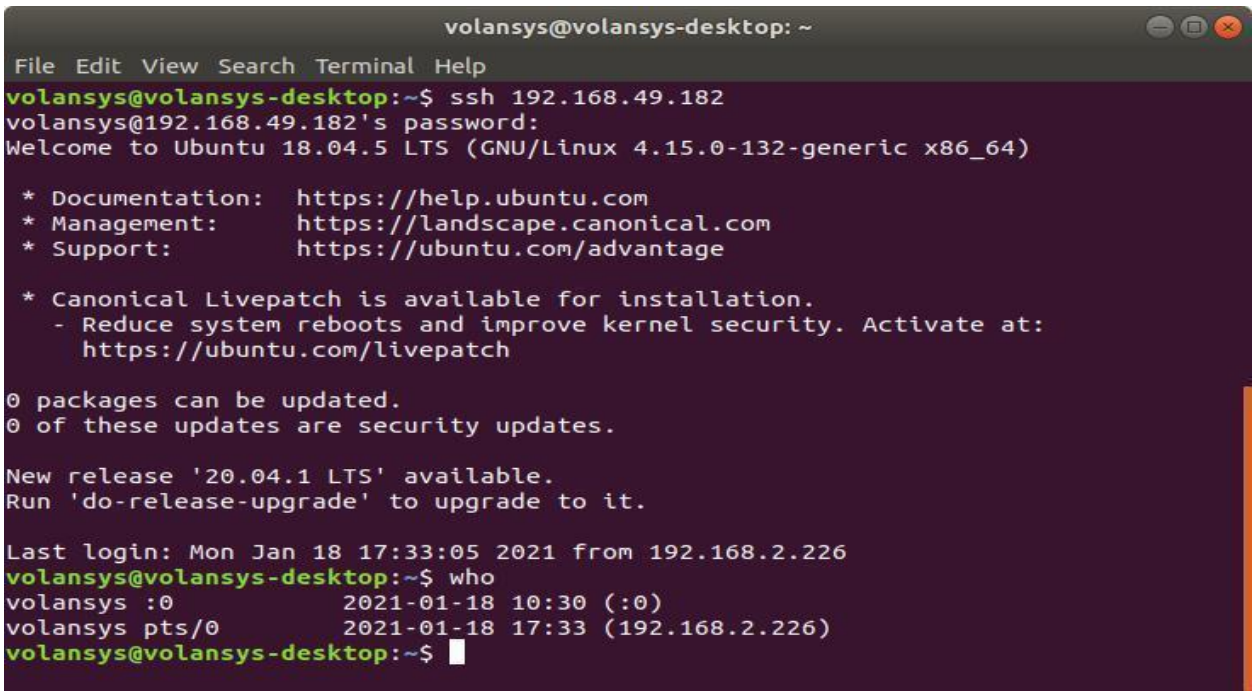


Fig.1.34 output of ssh command

SCP command

==>to copy the file from the remote system.

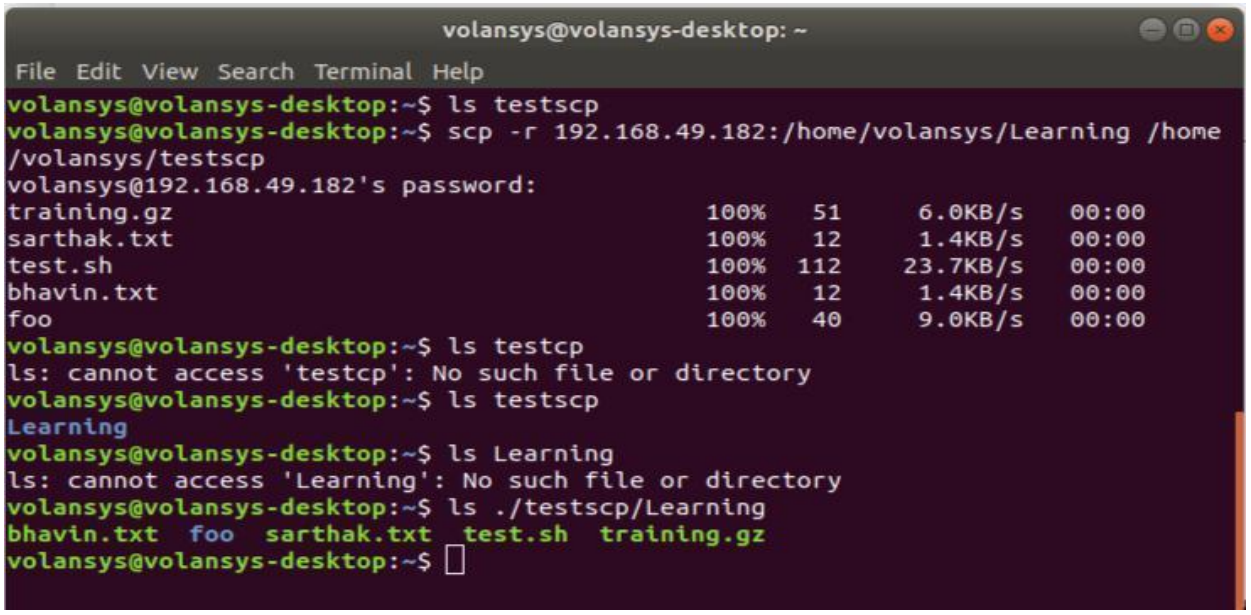


Fig.1.35 output of scp command

1.13 Advance file-system mounting

1.13.1 Samba

shareable folder using samba:

==>

Step:1

Make a directory and change its permissions.

Step:2

To make the directory shareable add its detail in /etc/samba/smb.conf file as mentioned below

```
[samba_share]
        Comment = Samba Shared Directory
        path = /home/volansys/samba
        writable = yes
        guest ok = yes
        read only = no
        force user = nobody
```

Step: 3

Now restart the smbd service.

Step: 4

access the shareable folder of the remote machine.

```
volansys@volansys-desktop:~$ sudo smbclient -L 192.168.2.24
WARNING: The "syslog" option is deprecated
Enter WORKGROUP\root's password:

        Sharename       Type      Comment
        ---------       ----      -------
        print$          Disk      Printer Drivers
        samba           Disk      samba share
        IPC$            IPC       IPC Service (volansys-desktop server (Samba, U
buntu))
        HP_LaserJet_Pro_MFP_M226dw_AAED7A_ Printer   HP LaserJet Pro MFP M226dw
(AAED7A)
Reconnecting with SMB1 for workgroup listing.

        Server               Comment
        ---------            -------

        Workgroup            Master
        ---------            -------
        WORKGROUP
```

```
volansys@volansys-desktop:~$ /usr/bin/smbclient //192.168.2.24/samba 'vt123'
WARNING: The "syslog" option is deprecated
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Mon Jan 18 15:46:45 2021
  ..                                  D        0  Mon Jan 18 15:46:13 2021
  foo.txt                             N        0  Mon Jan 18 15:46:45 2021
  swapnil                             D        0  Mon Jan 18 15:46:42 2021

                105210648 blocks of size 1024. 89395708 blocks available
```

Fig.1.36 accessing the shared directory of remote machine

1.13.2 NFS

shareable folder using nfs:

Step:1

Make a directory and change its permissions.

Step:2

To make the directory shareable, edit /etc/exports file as mentioned below.

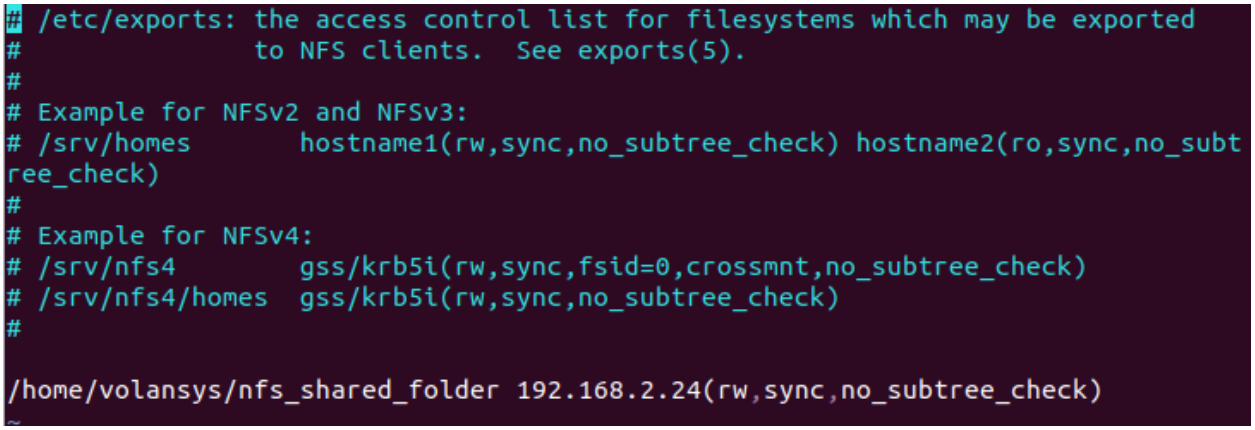


Fig.1.37 editing /etc/exports file for making directory sharable using nfs.

Step:3

Now export the shared directory using command sudo exportfs -a

Step:4

Now restart the nfs service.

1.14 Shell Script Basics

shell script

Shell script is simply a text file which contains a sequence of commands and has a .sh extension.

Purpose

Suppose we have some task and to do that task we need to execute multiple commands in sequence. So whenever we want to do that task we need to execute all those commands one by one every time.

Instead of doing this we can write all these commands into one text file and save with .sh extension.

After that make that file executable and also add the path of its directory to the PATH variable.

Now whenever we want to do that task we need to execute only one file which is our shell script and this can be done by ./filename.

So, it is basically used to automate certain tasks and to save our time.

Chapter 2: Gitlab Training

2.1 Version Control System

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

- **Functions of a VCS:**
    1. Allows developers to work simultaneously.
    2. Does not allow overwriting each other's changes.
    3. Maintaining history of every version.

- **Types of VCS:**
    1. Centralized version control system (CVCS).
    2. Distributed/Decentralized version control system (DVCS).

- **Centralized Version Control System(CVCS)**

  Centralized version control system (CVCS) uses a central server to store all files and enables team collaboration. But the major drawback of CVCS is failure of the central server. Unfortunately, if the central server goes down for an hour, then during that hour, no one can collaborate at all. And even in a worst case, if the disk of the central server gets corrupted and proper backup has not been taken, then you will lose the entire history of the project.

- **Distributed Version Control System(DVCS)**

  These DVCS systems do not necessarily rely on a central server to store all the versions of a project file.

  In Distributed VCS, every contributor has a local copy or "clone" of the main repository i.e. everyone maintains a local repository of their own which contains all the files and metadata present in the main repository.You can commit changes, create branches, view logs, and perform other operations when you are offline. You require network connection only to publish your changes and take the latest changes.

2.2 Workflow of GIT

Step 1 : modify a file from the working directory.

Step 2 : add these files to the staging area.

Step 3 : perform a commit operation that moves the files from the staging area. After push operation, it stores the changes permanently to the GIT repository.

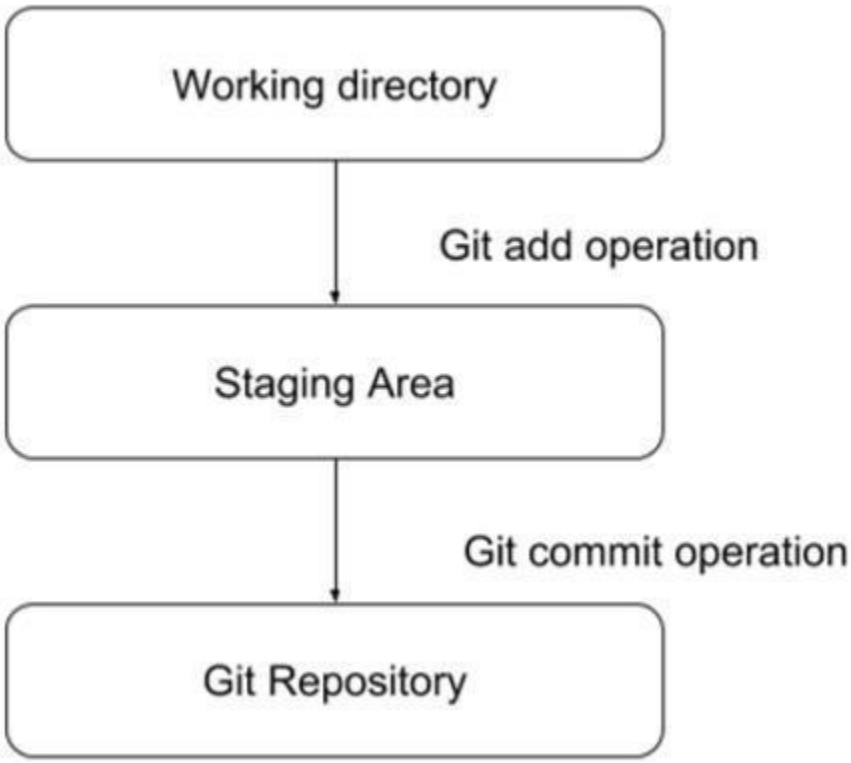


Fig.2.1 git workflow

2.3 Branch in Gitlab

Master branch: when our product version is ready to release then only we should commit on master branch because every commit on master branch is known as version of product and after committing we should create a tag so that nobody can change that version.

Develop branch: when we want to make a new version of an already existing product then we can make that next version on develop branch. It means every commit on the development branch is the latest development change for the next version and when the development branch reaches the point when it becomes stable and ready to release then we can merge the development branch with the main branch and release our next version.

Feature branch: when we develop the next version of a product on develop branch then we need to develop several new features. For all these features we make one dedicated feature branch which branches off from develop branch and merge with develop branch itself.

Release branch: when develop branch acquire enough features for reales then release branch will branch off from develop branch after that develop branch can not develop any new feature until next release. In release branch bug fixing and other tasks related to release will be done. Once it is ready it will merge with master and develop branches.

Hotfix branch: basically hotfix branch is used for quick release of the next version. Suppose we want some bug fixing and some extra features in the next release. Suppose Bug fixing will take less time and development of features will take much more time. In this case we can use develop branch for developing new release with required feathers and hotfix branch for bug fixing which will branch off master branch and merge with master and develop branch. As a result bug fixing will be done earlier and it will merge with master. It means it will release the next version quickly with bug fixing and also merge with develop so in the next release we can combine bug fixing along with new features.

2.4 Basic Gitlab command

git clone ==> clone git repository on your machine

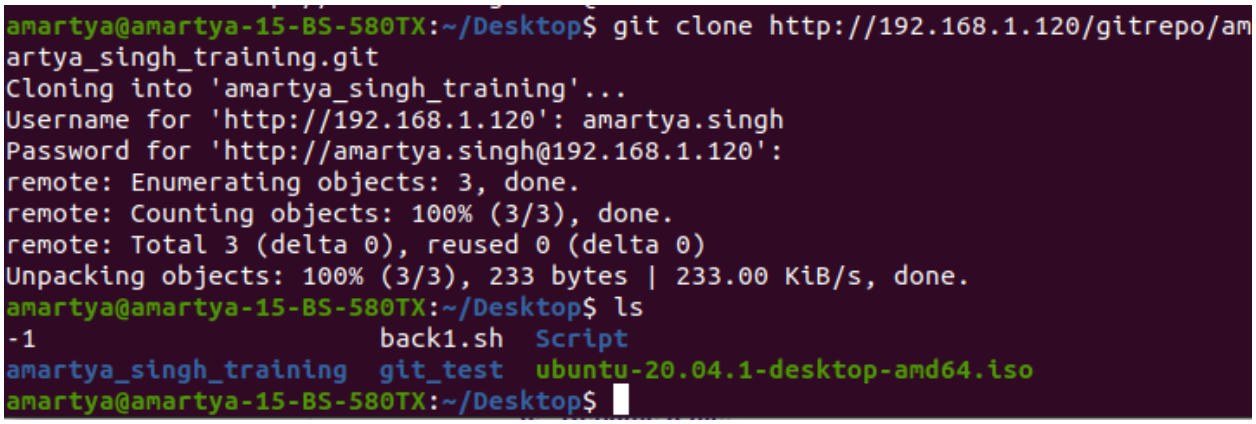


Fig.2.2 Cloning git repository with http

git add ==> add file from workspace to staging area
git commit ==> add file from staging area to local repository.

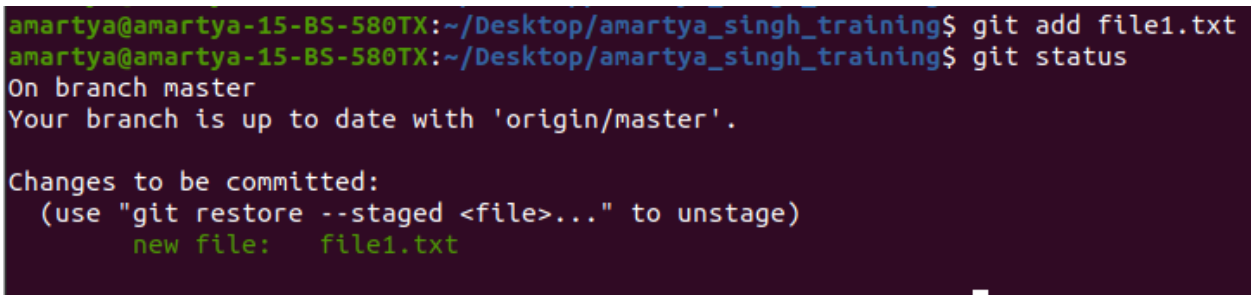


Fig.2.3 add a file to staging area and to local repository.

git push ==> push the changes for local repository to git repository

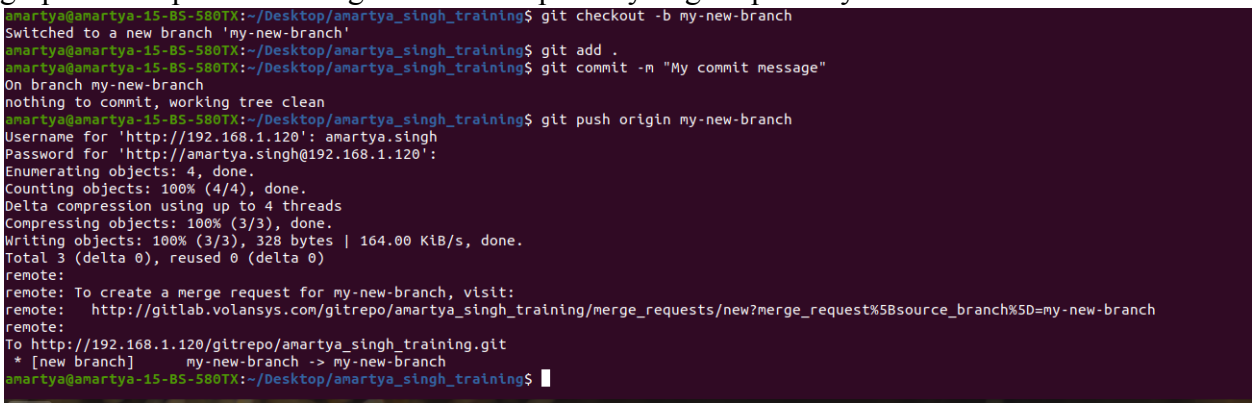
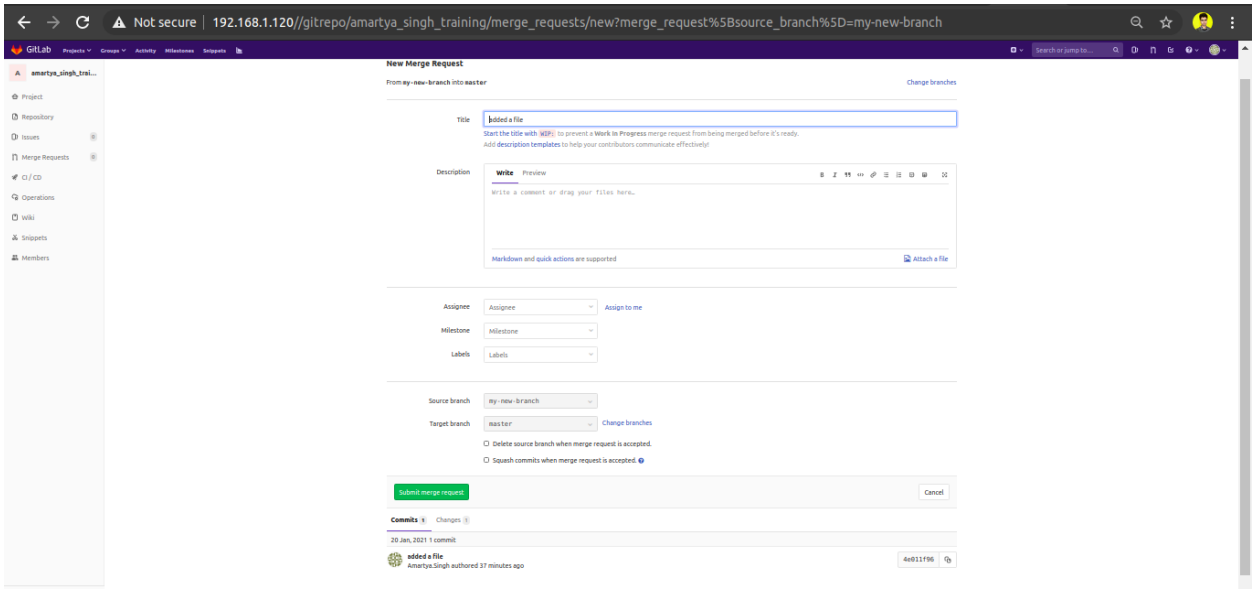


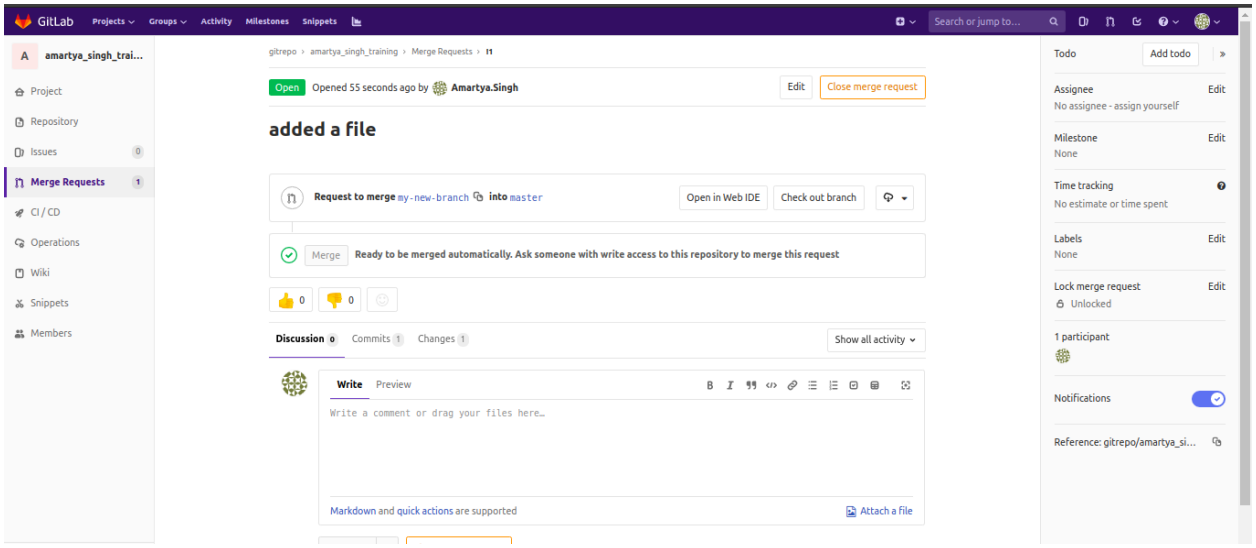
Fig.2.4 push changes from local repository to git repository

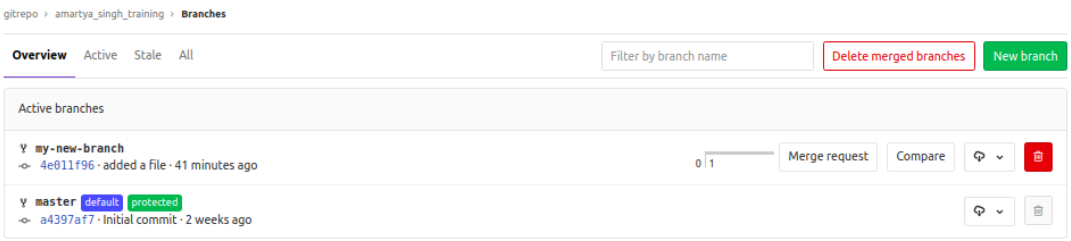
git push origin my-new-branch

The output of this command in the above question produced a link to create a merge request. If the merge request is approved then you will be also able to push changes to the master branch.



Creating merge request





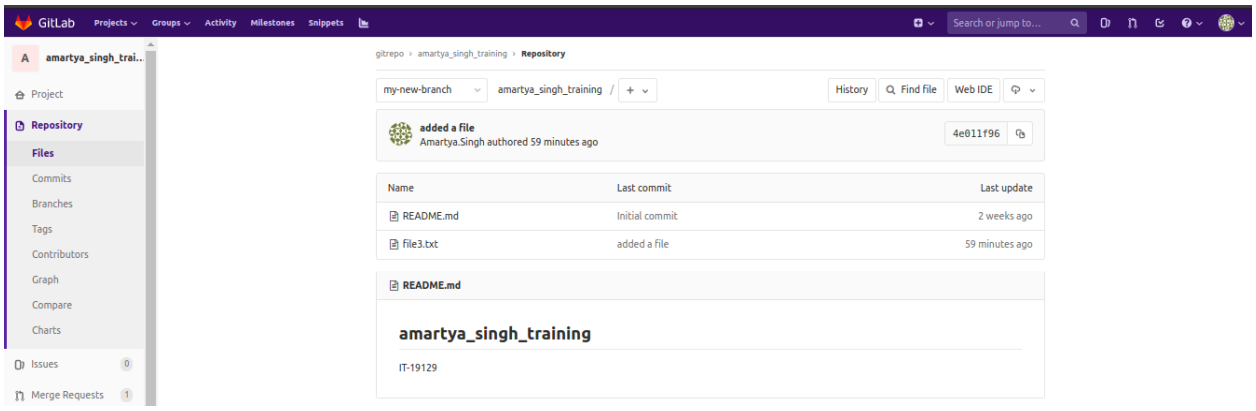


Fig.2.5 Create a Merge Request

Chapter 3 C Programming Training

3.1 Doxygen

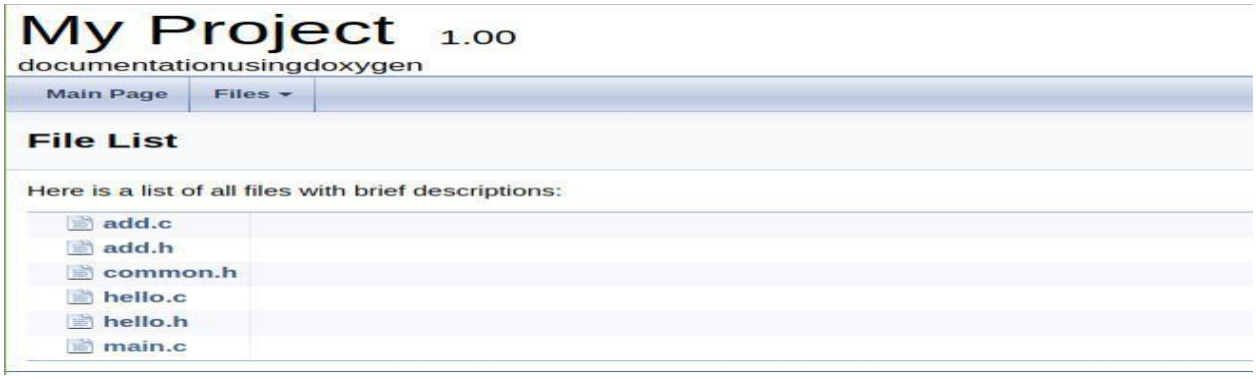
3.1.1 What is Doxygen?

Doxygen is a software used to produce documentation of source code written in C, C++, Python, Java, etc. and delivers in various formats like HTML, PDF, etc.

3.1.2 How Doxygen works?

Doxygen works by taking the comments which are specifically formatted according to Doxygen's syntax, when you run this tool, it will parse out the documentation details from comments which follow the special Doxygen syntax. so that they can be displayed in formats like HTML, Hyperlinked PDF, etc.

3.1.3 Example



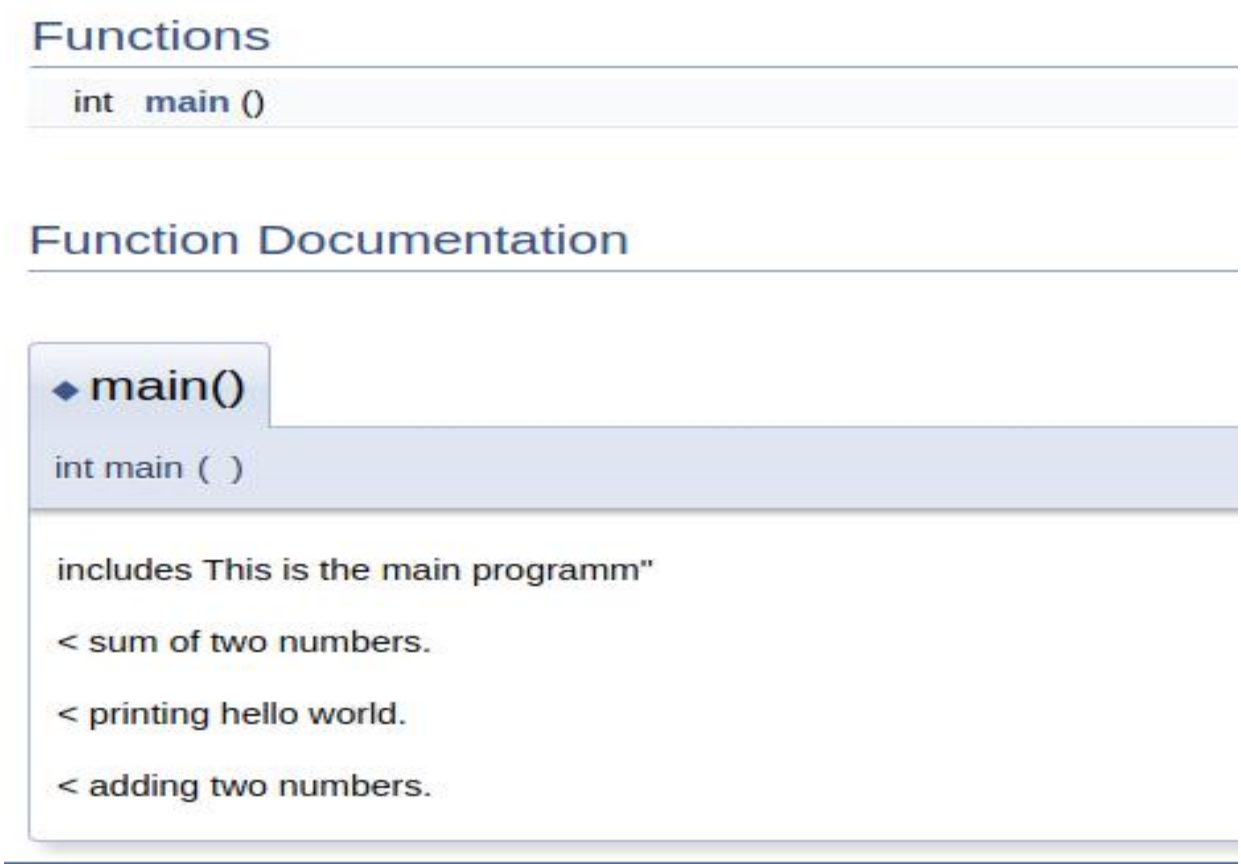


Fig.3.1 example of doxygen

3.2 segments of the executable file

- Executable file has basically five segments.
    1. text/code segment =>this segment contains source codes of program.
    2. Data segment =>this segment contains initialized global and static variables.
    3. bss(block starting symbols) segment=>this segment contains uninitialized global and static variables.
    4. Stack segment=>this segment contains local variables.
    5. Heap segment =>this segment contains dynamically allocated memory.

3.3 Compilation Stages

- **Compilation process is made by 4 steps:**

Step1: pre-processing

Pre-processor takes a .c file as an input and generates .i file.

In this step all the macros name will replace by its value in whole program , header filename is replaced with actual header file and other conditional compilations instructions like #ifdef, #ifndef, etc are processed.

Step2: compilation

Compiler takes a preprocessed file (.i file) as an input and generates an assembler source file (.sfile).

Step3: assembling

Assembler takes assembler source file (.s file) as an input and generates objectfile (.obj file)

Step4: linking

Linker takes one or more object files as an input and generates an executable file (.exe file).

3.4 Debugging using gdb command

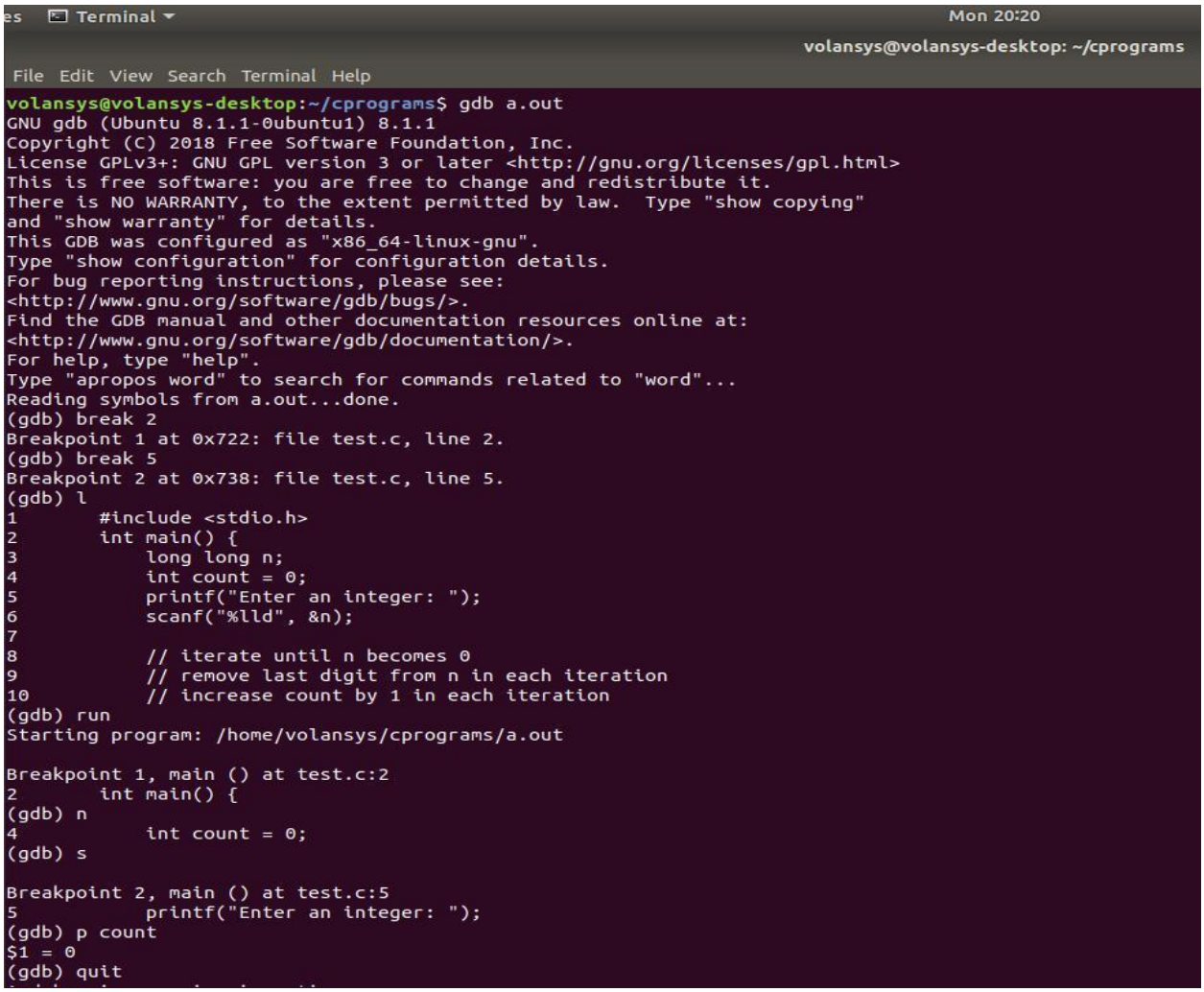


Fig.3.2 output of gdb command

3.5 Reading of basics of c programming language

Revising the basic concepts of c programming language like variables, data types, operators, typedef, type casting and storage class, control flow, decision making, looping, functions and scope rules, basic pointers, array, structure, unions, bit field, enumeration, function pointers, Input and output.

3.6 Custom malloc and free function

In this exercise I have developed a custom malloc and custom free function which acts as an actual malloc and free function and for that I have declared an array which is acting as heap memory. Initially the whole heap is initialized as a meta data block and as a data block. When we call malloc function it will check for available memory and if memory is available then creates one metadata block and one data block and allocates this block by returning its starting address. Similarly a free function will release the block allocated by malloc function.

Output:

```
volansys@volansys-Desktop:~/Training/C_Programming/Mega_Excerise/Part_1/Sources$ ./malloc_free
Inside: malloc(20):
Inside: malloc(256):
Inside: malloc(20):

Memory gets freed successfully for a

Memory gets freed successfully for b
```

Fig.3.3 custom malloc and free function

Chapter 4 Advanced C Training

4.1 The Valgrind

4.1.1 Introduction

The Valgrind tool suite provides a number of debugging and profiling tools that help you make your programs faster and more correct. The most popular of these tools is called Memcheck. It can detect many memory-related errors that are common in C and C++ programs and that can lead to crashes and unpredictable behaviour.

4.1.2 Preparing your program

Compile your program with -g to include debugging information so that Memcheck's error messages . Using different flags we can get the line number , where the error has occurred , leaks we can detect and many more.

4.1.3 Running your program under Memcheck

If you normally run your program like this:

==> a.out

Use this command line:

==> valgrind --tool=memcheck --leak-check=yes a.out

The --leak-check option turns on the detailed memory leak detector.

4.1.4 Example

```c
#include <stdio.h>
#include <stdlib.h>

void f(void)
{
  int* x = malloc(10 * sizeof(int));
  x[10] = 0;          // problem 1: heap block overrun
}                     // problem 2: memory leak -- x not freed

int main(void)
{
  f();
  return 0;
}
```

Fig.4.1 Code with errors

```
volansys@volansys-Desktop:~/Training/Advanced_C$ valgrind ./a.out
==4724== Memcheck, a memory error detector
==4724== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==4724== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==4724== Command: ./a.out
==4724==
==4724== Invalid write of size 4
==4724==    at 0x108668: f (valgrind.c:7)
==4724==    by 0x108679: main (valgrind.c:12)
==4724==  Address 0x522f068 is 0 bytes after a block of size 40 alloc'd
==4724==    at 0x4C31B0F: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==4724==    by 0x10865B: f (valgrind.c:6)
==4724==    by 0x108679: main (valgrind.c:12)
==4724==
==4724==
==4724== HEAP SUMMARY:
==4724==     in use at exit: 40 bytes in 1 blocks
==4724==   total heap usage: 1 allocs, 0 frees, 40 bytes allocated
==4724==
==4724== LEAK SUMMARY:
==4724==    definitely lost: 40 bytes in 1 blocks
==4724==    indirectly lost: 0 bytes in 0 blocks
==4724==      possibly lost: 0 bytes in 0 blocks
==4724==    still reachable: 0 bytes in 0 blocks
==4724==         suppressed: 0 bytes in 0 blocks
==4724== Rerun with --leak-check=full to see details of leaked memory
==4724==
==4724== For counts of detected and suppressed errors, rerun with: -v
==4724== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

Fig.4.2 running program with valgrind

4.2 Makefile

4.2.1 Overview of make

Make which pieces of the program need to be recompiled, and fire commands to recompile them.

4.2.2 Preparing and Running Make

Write a file named makefile that tells the relationship among other files in the program and fires command hence updating each file. Executables file is updated from object files, which are made by compiling source files.

Now each time we change some source file, fire this command

make

perform all necessary steps related to compilation. Make programs use the makefile database and last-modification time log of the file to decide which of the file need to be compiled.

4.2.3 Example

```
# define the C compiler to use
CC = gcc

# define any compile-time flags
CFLAGS = -Wall -g

# define any directories containing header files other than /usr/include
#
INCLUDES = ../Includes

# define the C source files
SRCS := $(wildcard *.c)

# define the C object files
#
# This uses Suffix Replacement within a macro:
#   $(name:string1=string2)
#         For each word in 'name' replace 'string1' with 'string2'
# Below we are replacing the suffix .c of all words in the macro SRCS
# with the .o suffix
#
OBJS = $(SRCS:.c=.o)
# define the executable file
MAIN = snake_game

#
# The following part of the makefile is generic; it can be used to
# build any executable just by changing the definitions above and by
# deleting dependencies appended to the file from 'make depend'
#

.PHONY: clean

all: $(MAIN)
        @echo  Snake Game has been compiled

$(MAIN): $(OBJS)
        $(CC) $(CFLAGS) -o $@ $^

clean:
        $(RM) *.o *~ $(MAIN)
```

Fig.4.5 makefile

4.3 Memory organization

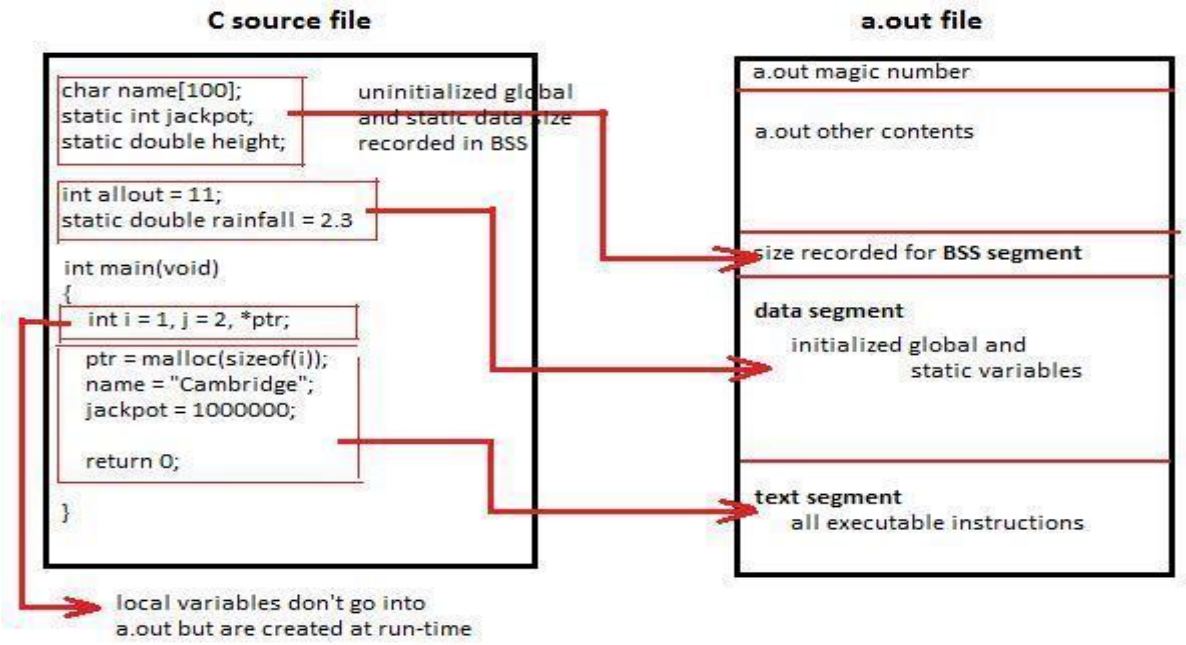


Fig.4.6 mapping of source file and executable file

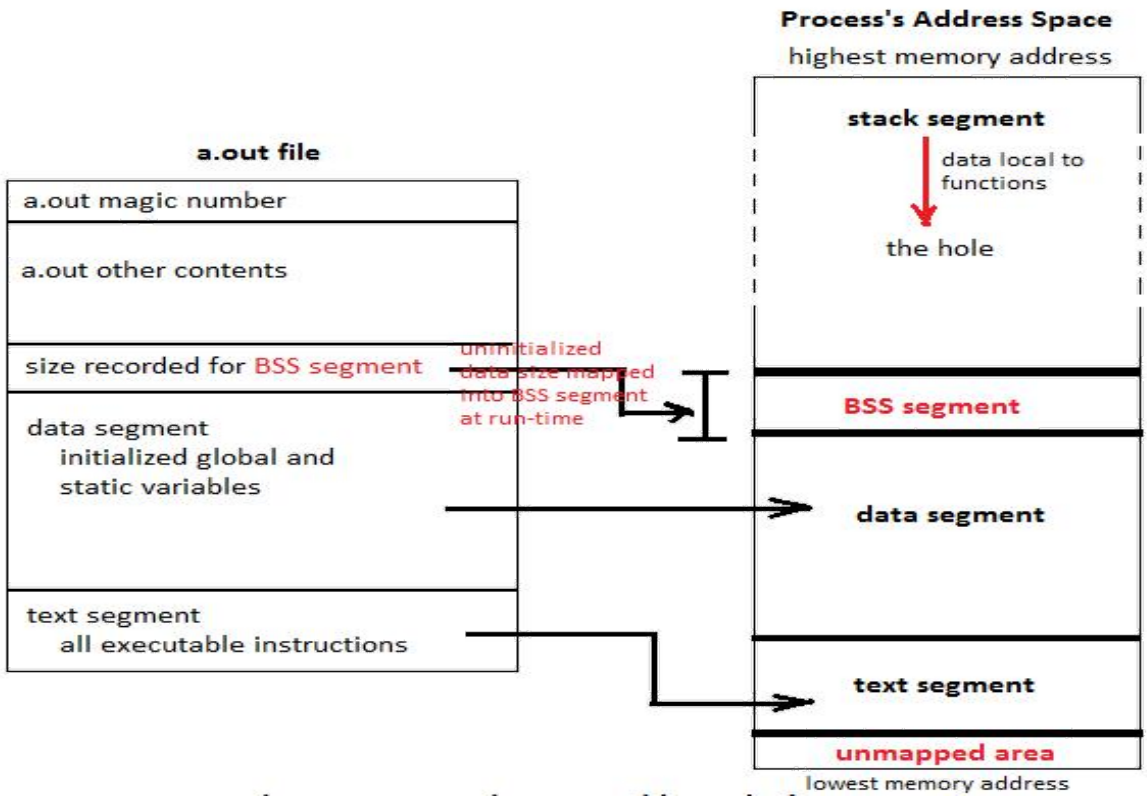


Fig.4.7 mapping of executable file and memory segments

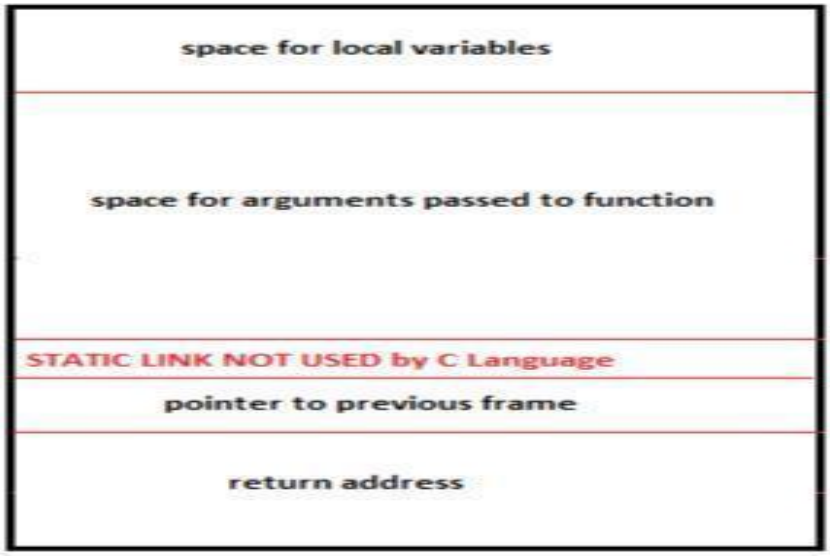


Fig.4.8 function activation record or stack frame

Chapter 5: Data Structures and Algorithms

5.1. Linked list

Linked List is a chain of link which contains some content. Each link contain address to another link.

- Node − Each Node of a linked list can store data.
- Next − Each Node of a linked list contains an address to the next Node.
- HEAD − points to the first node of the linked list.

5.1.1 Linked list Representation

5.1.2 Types of Linked List

- Single Linked List −Only points to next or previous node.
- Double Linked List − Points to next & previous node.
- Circular Linked List − Tail element points head as next element.

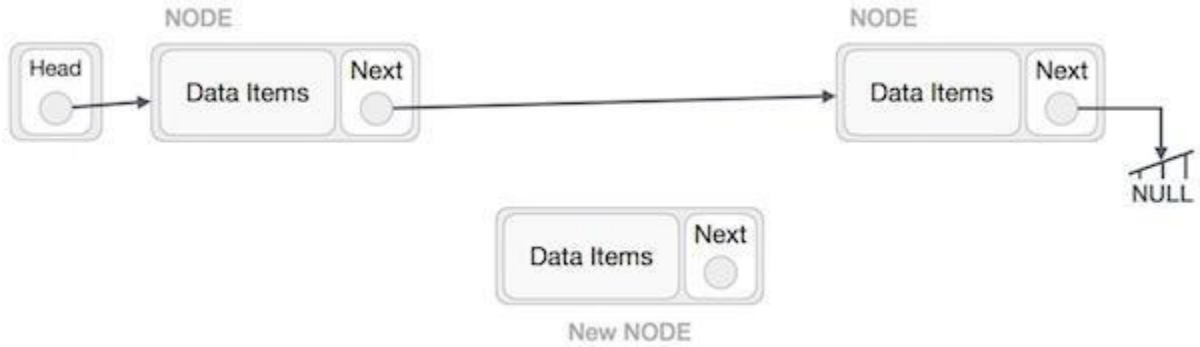
5.1.3 Basic Operations

Following are the basic operations supported by a list.

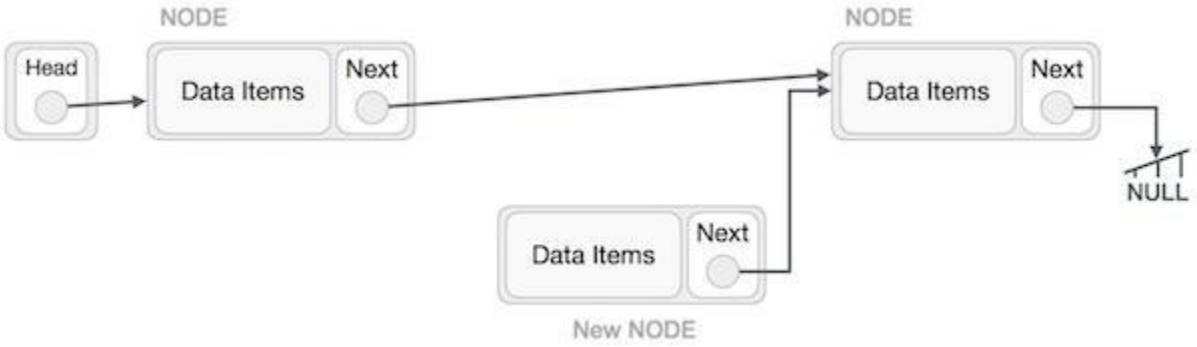
- Insert − Add node at the beginning or end of linked list
- Delete − Delete node at the beginning or end of linked list
- Traverse − Going through the linked list.
- Search − Search given element in the list.

Insertion Operation

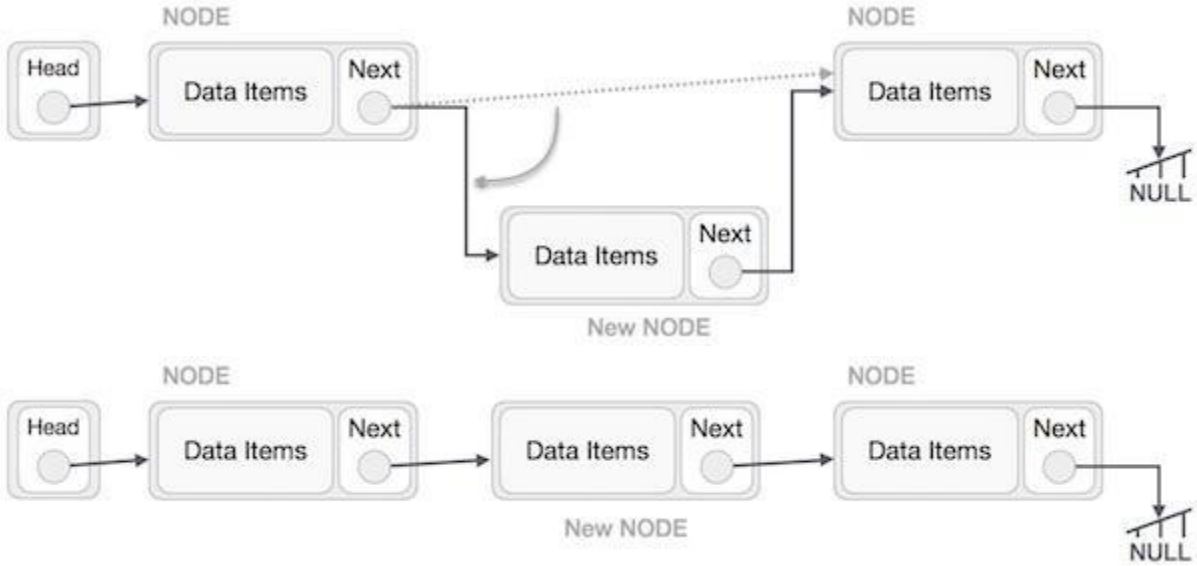
Create a node using the structure and find location where it will be inserted.



newNode.next−>rightNode;



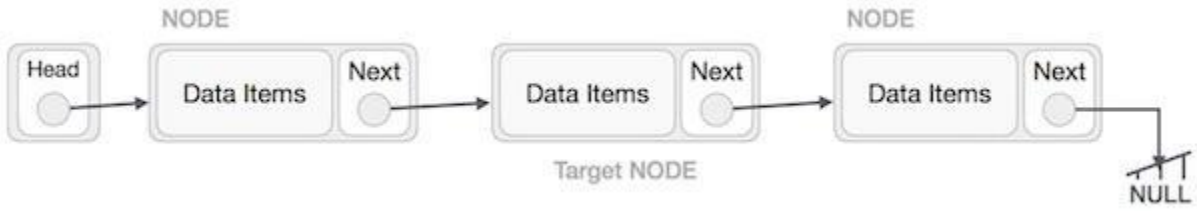
leftNode.next−>newNode;



When inserting it at the last, the second last node of the list points to the new node and the new node points to NULL.

Deletion Operation

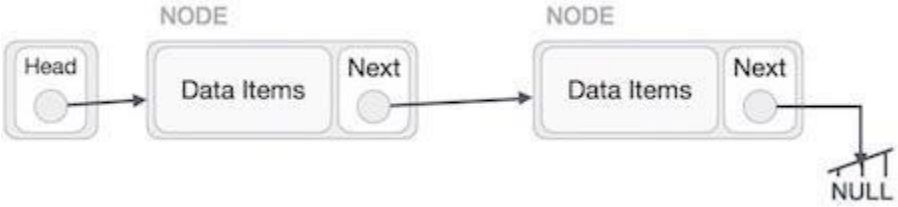
Locate the target node to be removed, by using any search algorithm.



leftNode.next−>targetNode.next;

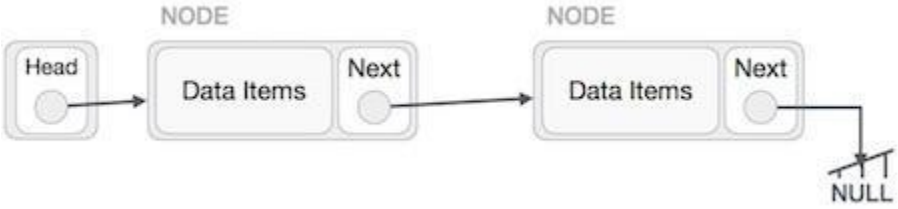
This removes the link that was pointing to target node.

targetNode.next −> NULL;

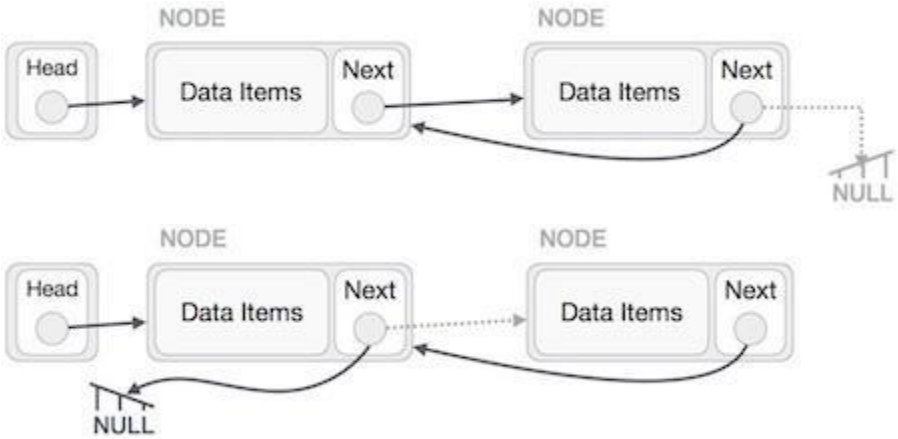


Reverse Operation

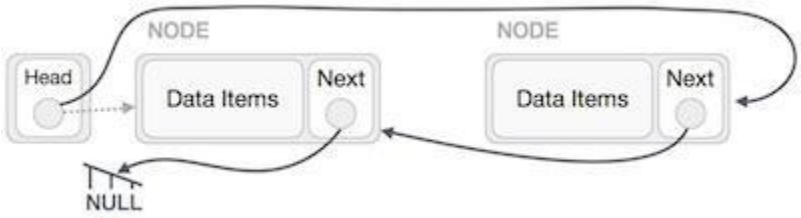
Last node will be pointed by head node and reverse the whole linked list.



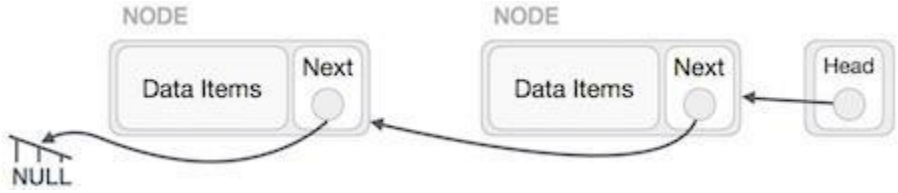
Traversing to the end of the list. It should be point NULL. Making it point to its previous node −



Except first node pointed by the head node, all other node should point to their predecessor, making them their new successor. The first node will point to NULL.



Making the head node point to the new first node by using the temp node.



5.2. Binary Search Tree

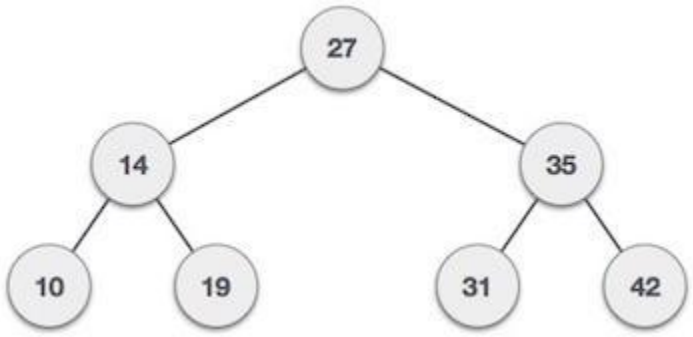
Binary Search Tree properties −

- Value of key of left subtree is less than the value of its parent (root) node key.
- Value of the key of the right subtree is more than or equal to the value of its parent (root) node key.

Thus, BST divides all its sub-trees into two segments; the left subtree and the right subtree and can be defined as −

left_subtree (keys) < node (key) ≤ right_subtree (keys)

5.2.1 Binary Search Tree Representation



Observe that the root node key (27) has all less valued key on the left subtree and the higher valued key on the right subtree.

5.2.2 Basic Operations

Basic operations of a tree −

- o Search − Search element in tree.

- Insert − Insert element in tree.
- Preorder Traversal − Traverse tree in preorder manner.
- Inorder Traversal − Traverse tree in inorder manner.
- Postorder Traversal − Traverse a tree in a postorder manner.

Chapter 6: BLE TLV Frame Emulator

6.1. System Overview

This project will simulate a scenario in which the set and get commands are sent from a mobile (User input on terminal in our case) and get appropriate response from the device (response will be generated from the users system in our case) in BLE command frame (TLV) format. In short this product will be a TLV frame emulator.

6.2. Theory Of Operation

Users will need to manually enter the BLE command frame (TLV) on the terminal acting as a mobile device. Device response will be simulated by the users system only.

A TLV frame from the terminal will be sent to the system for set or get command then an appropriate response will be generated in the TLV structure itself in the users system and will be sent as a response to the user on the terminal.

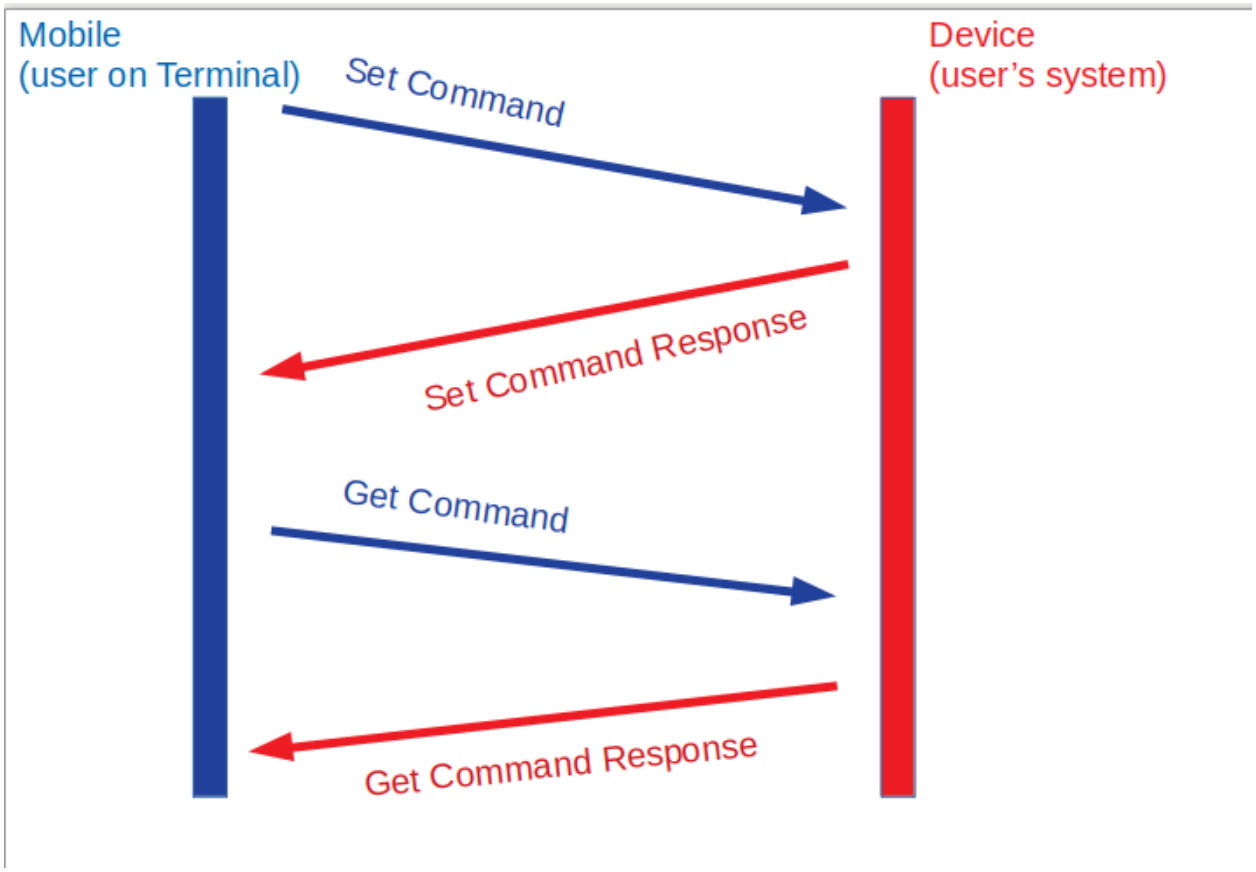


Fig.6.1: Flow of BLE

6.3. User Interface

```
volansys@volansys-desktop:~/Development/Project_Beta/source$ ./test
Input String (In Hexadecimal): 55AA600C620164630132640400007530
Response Frame Output: 55AA0F026000
Input String (In Hexadecimal): 55AA6100
Response Frame Output: 55AA610C620164630132640400007530
Input String (In Hexadecimal):
```

Fig.6.2: Output of BLE Header

6.4. System Architecture

This section is to describe top level architecture of the system. This is similar to high level design.

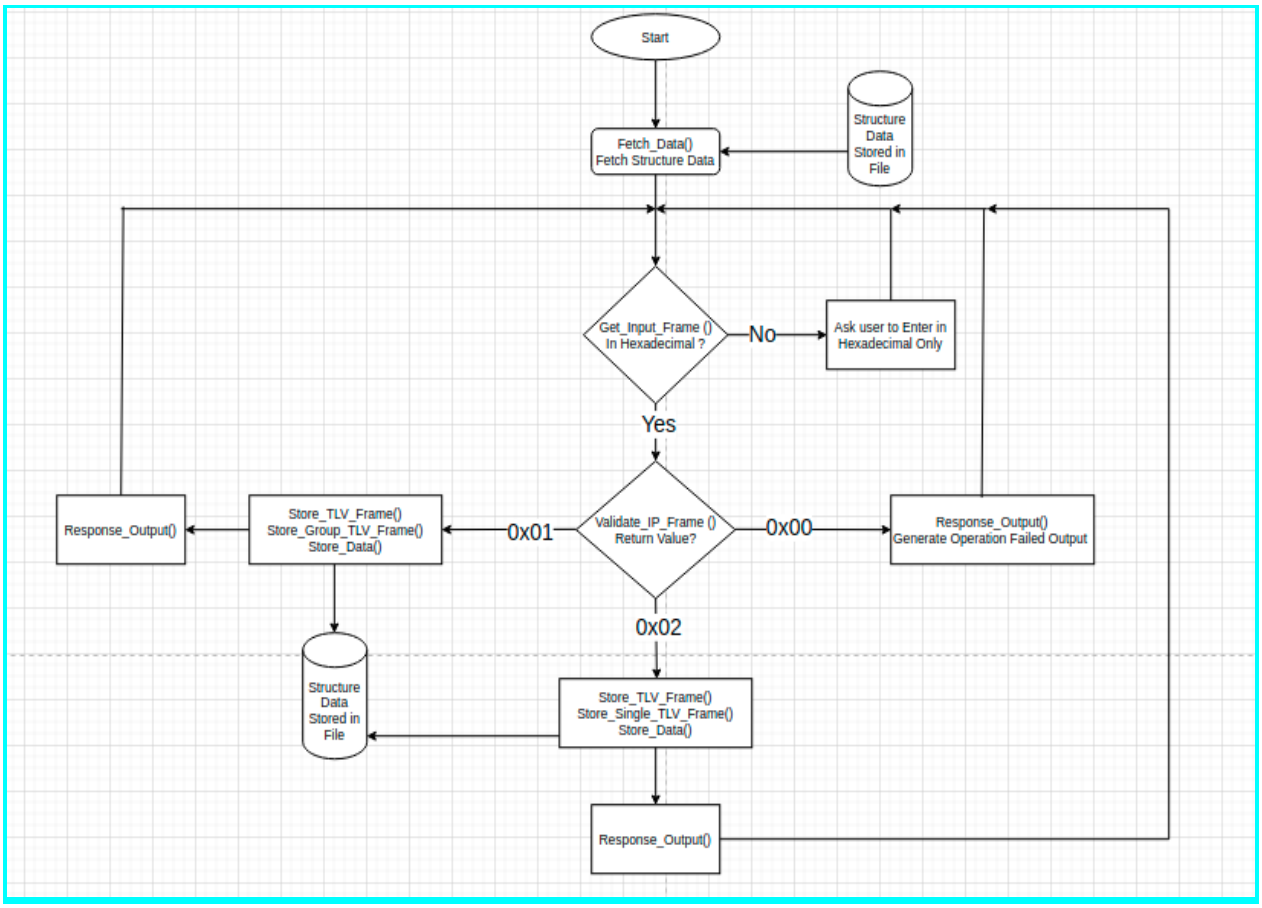


Fig.6.3: Flow of Project

Fetch_Data()

It will fetch the data of structures from the file.

Get_Input_Frame()

It will take input as a character array then convert it to hex array for further processing and also handles the inappropriate input for the program.

Validate_IP_Frame()

This will validate the template of the input array received if it's correct then it will process it further for range validation

Response_Output()

generate appropriate output for the user as per set and get command.

Store_TLV_Frame()

Stores the received input array in TLV structure.

Store_Single_TLV_Frame()

Stores the value in frame structure for single command

Store_Group_TLV_Frame()

Stores the value in frame structure for group command

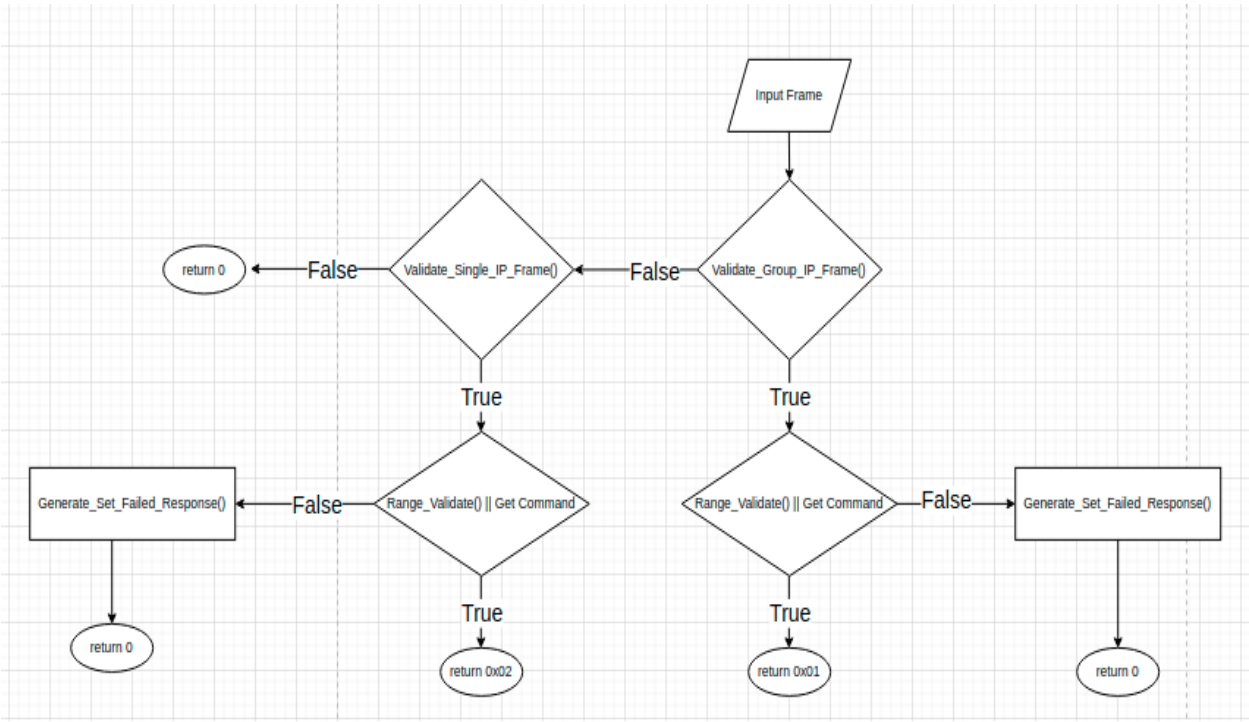


Fig.6.4: Validate IP Frame Function

Validate_Group_IP_Frame()

Verifies the input with the template of Group Frame.

Validate_Single_IP_Frame()

Verifies the input with the template of Single Frame.

Range_Validate()

Verifies the range as per the command id in the input frame.

Genrerate_Set_Failed_Response()

Generate failed response for the set command as range requirements are not matched.

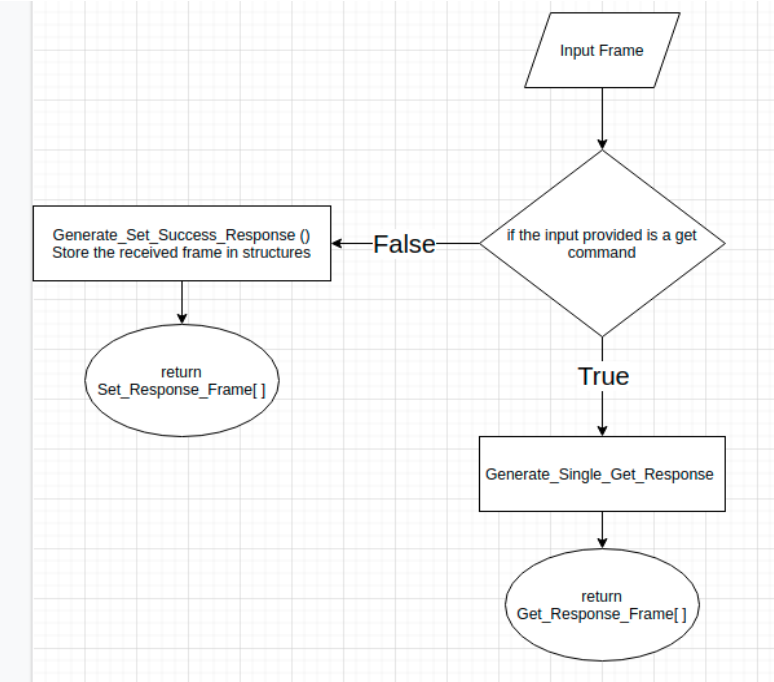


Fig.6.5: Store Single TLV Frame Function

Genrerate_Single_Get_Response()
Generate response frame for single get command.
Generate_Set_Success_Response()
Generate Set command success frame.
Set_Response_Frame and Get_Response frame are generated in this module as per the requirement of the received frame.

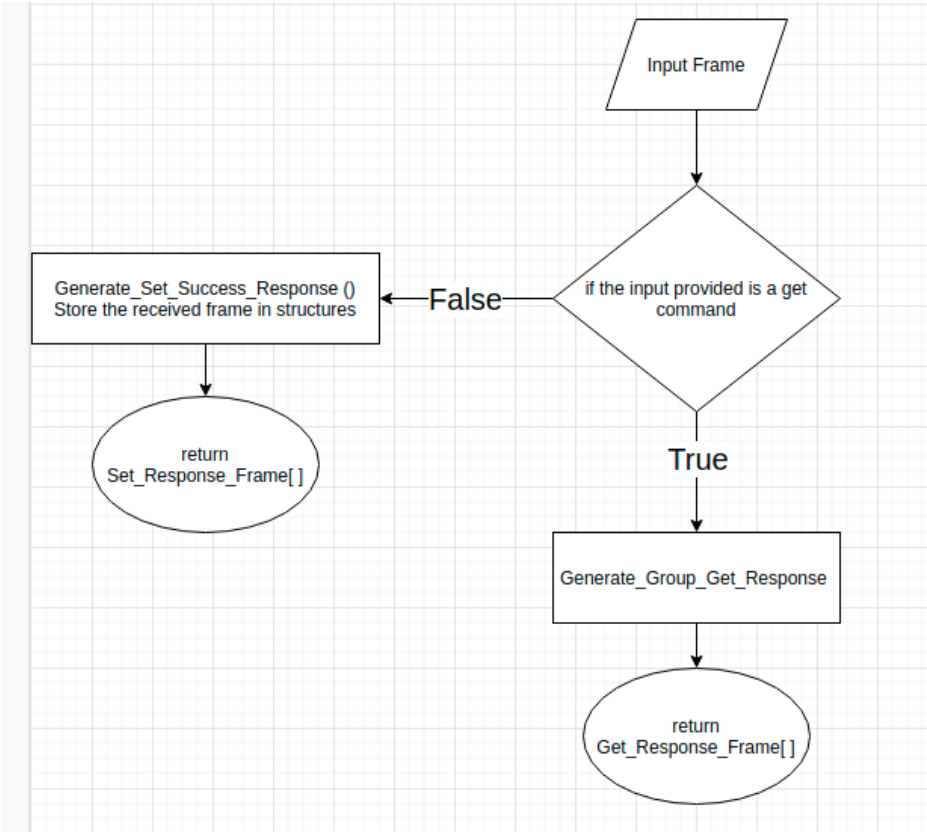


Fig.6.6: Store Group TLV Frame Function

Genrerate_Set_Success_Response()

Generate response frame for successful set command.

Generate_Group_Get_Response()

Generate the get command response for group command.

6.5. Functional Requirement

Req. ID	Requirement description
SR_2	Program should handle invalid input and should not crash if any invalid input is received.
SR_3	Program should also carry out range validation for the value in the received valid tlv frame.
SR_4	Program should store values received in TLV frame in proper structures for further use.
SR_5	Emulator Shall be Able to Generate appropriate response as per (set/get) Advertising Configuration Group TLV Frame
SR_6	Emulator Shall be Able to Generate appropriate response as per (set/get) Advertising Timeout Group TLV Frame
SR_7	Emulator Shall be Able to Generate appropriate response as per (set/get) Accelerometer Configuration Group TLV Frame
SR_8	Emulator Shall be Able to Generate appropriate response as per (set/get) Device Configuration Group TLV Frame
SR_9	Emulator Shall be Able to Generate appropriate response as per (set/get) Impact Configuration Group TLV Frame
SR_10	Emulator Shall be Able to Generate appropriate response as per (set/get) HIE configuration Group TLV Frame

SR_11	Emulator Shall be Able to Generate appropriate response as per (set/get) Advertising Configuration Group TLV Frame
SR_12	Emulator Shall be Able to Parse Advertising Timeout Group TLV Frame
SR_13	Emulator Shall be Able to Parse Accelerometer Configuration Group TLV Frame
SR_14	Emulator Shall be Able to Parse Device Configuration Group TLV Frame
SR_15	Emulator Shall be Able to Parse Impact Configuration Group TLV Frame
SR_16	Emulator Shall be Able to Parse HIE configuration Group TLV Frame
SR_17	Emulator Shall be Able to Parse Normal advertising frequency Single TLV Frame
SR_18	Emulator Shall be Able to Parse Check In advertising frequency Single TLV Frame
SR_19	Emulator Shall be Able to Parse HIE advertising frequency Single TLV Frame
SR_20	Emulator Shall be Able to Parse Alert advertising frequency Single TLV Frame
SR_21	Emulator Shall be Able to Parse Normal advertising timeout Single TLV Frame
SR_22	Emulator Shall be Able to Parse Check In advertising timeout Single TLV Frame
SR_23	Emulator Shall be Able to Parse HIE advertising timeout Single TLV Frame
SR_24	Emulator Shall be Able to Parse Alert advertising timeout Single TLV Frame
SR_25	Emulator Shall be Able to Parse Accelerometer scale Single TLV Frame
SR_26	Emulator Shall be Able to Parse Accelerometer Wakeup Threshold Single TLV Frame
SR_27	Emulator Shall be Able to Parse Emfit Sensing Threshold Single TLV Frame
SR_28	Emulator Shall be Able to Parse Emfit Sample Rate Single TLV Frame
SR_29	Emulator Shall be Able to Parse Emfit Total Sample Single TLV Frame
SR_30	Emulator Shall be Able to Parse Check in interval Single TLV Frame
SR_31	Emulator Shall be Able to Parse Inactive Timeout Single TLV Frame
SR_32	Emulator Shall be Able to Parse HIE Queue size Single TLV Frame
SR_33	Emulator Shall be Able to Parse HIE Threshold Single TLV Frame
SR_34	Emulator Shall be Able to Parse HIE Timeout Single TLV Frame
SR_35	Emulator Shall be Able to Generate appropriate response as per (set/get) Normal advertising frequency Single TLV Frame
SR_36	Emulator Shall be Able to Generate appropriate response as per (set/get) Check In advertising frequency Single TLV Frame
SR_37	Emulator Shall be Able to Generate appropriate response as per (set/get) HIE advertising frequency Single TLV Frame
SR_38	Emulator Shall be Able to Generate appropriate response as per (set/get) Alert advertising frequency Single TLV Frame
SR_39	Emulator Shall be Able to Generate appropriate response as per (set/get) Normal advertising timeout Single TLV Frame
SR_40	Emulator Shall be Able to Generate appropriate response as per (set/get) Check In advertising timeout Single TLV Frame
SR_41	Emulator Shall be Able to Parse HIE advertising timeout Single TLV Frame
SR_42	Emulator Shall be Able to Generate appropriate response as per (set/get) Alert advertising timeout Single TLV Frame
SR_43	Emulator Shall be Able to Generate appropriate response as per (set/get) Accelerometer scale Single TLV Frame
SR_44	Emulator Shall be Able to Generate appropriate response as per (set/get) Accelerometer Wakeup Threshold Single TLV Frame
SR_45	Emulator Shall be Able to Generate appropriate response as per (set/get) Emfit Sensing Threshold Single TLV Frame

SR_46	Emulator Shall be Able to Generate appropriate response as per (set/get) Emfit Sample Rate Single TLV Frame
SR_47	Emulator Shall be Able to Generate appropriate response as per (set/get) Emfit Total Sample Single TLV Frame
SR_48	Emulator Shall be Able to Generate appropriate response as per (set/get) Check in interval Single TLV Frame
SR_49	Emulator Shall be Able to Generate appropriate response as per (set/get) Inactive Timeout Single TLV Frame
SR_51	Emulator Shall be Able to Generate appropriate response as per (set/get) HIE Queue size Single TLV Frame
SR_52	Emulator Shall be Able to Generate appropriate response as per (set/get) HIE Threshold Single TLV Frame
SR_53	Emulator Shall be Able to Generate appropriate response as per (set/get) HIE Timeout Single TLV Frame

Table 6.1 Functional Requirement

Chapter 7: Snake Game

7.1. System Overview

The most popular classical game named “SNAKE” starts by welcoming on the screen, User is asked whether he/she is a new user or existing user and based on that the information is being stored and retrieved. After registering , the User is asked to select the game mode they want to play.After that, the game starts where the user controls a snake by using arrow keys until it manages to eat food. When you get the food, the Snake grows by an extra block .If, or rather when, the snake bumps into the edge of the boundary or accidentally eats himself the game is over. And the final Score , the User gets is shown.

7.2. Theory Of Operation

The Snake Game starts by reading the config file which includes mostly speed of snake , shape of snake , shape of food , Total Number of food and other things.After that a Welcome Message is shown on the prompt and asks the user whether he/she is new in this game or not.

If the user is new , then the user is asked to enter the user_id and name. After that the user is being asked to select the game level he/she wants to play . Game levels are Easy , Medium , Difficult . All these levels are categorized on the bases of the speed of the snake and the boundary area of the game. After selecting the game level , a prompt of game instructions is being shown to the user . After this the Game starts.

If the user is not new in this game , then the user is asked to enter the user_id , After that the Scoreboard is shown to the user , the Scoreboard contains the Top 5 Global player and the score of old games the user has played . After that the user is asked to select the game levels and then the game starts.

According to the game level the user has selected , the game is being initialized , and the maximum chances the user gets to play this game is three times after that the game is over. After Initializing , the snake is being put at a random location inside the boundary . The first food snake can get will be after some milliseconds the user had started to play the game. The user can control the movement of the snake by using the arrow keys. At each time the snake eats the food , the score is being incremented , if the snake touches the boundary or ate himself then the life is being deducted , When no life is there the Game is over and the score of the user is being shown and is being recorded in the file. If the user beatles its own score then a message of Congratulations is prompted. If the user wishes to play again , the game starts at selection of game level otherwise the game is being quit.

7.3. User Interface

Fig.7.1 User press ‘r’ for register

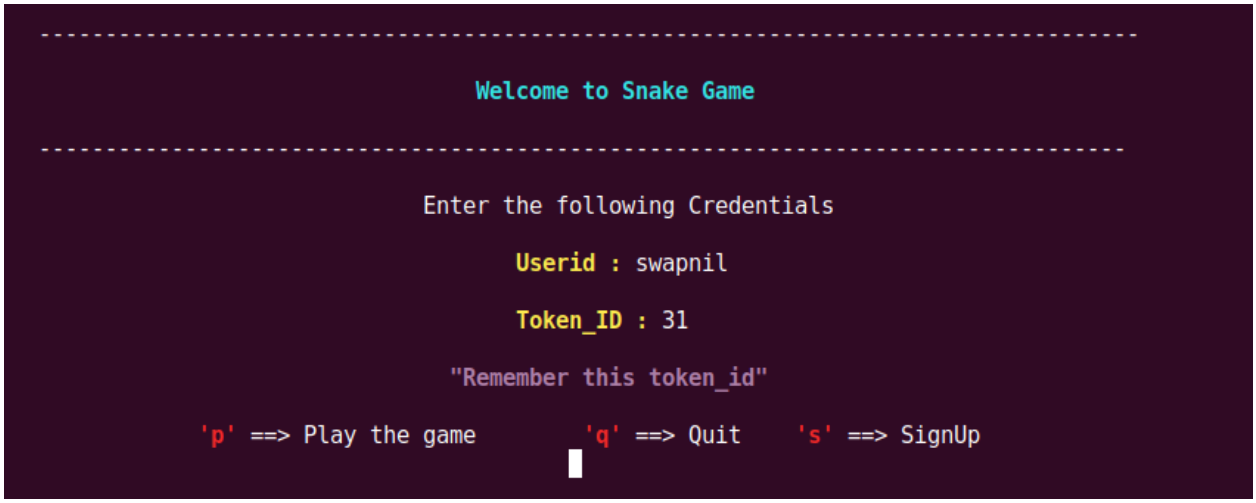


Fig.7.2 Register page and press ‘p’

Fig.7.3 Game Instructions & press ‘c’ to continue

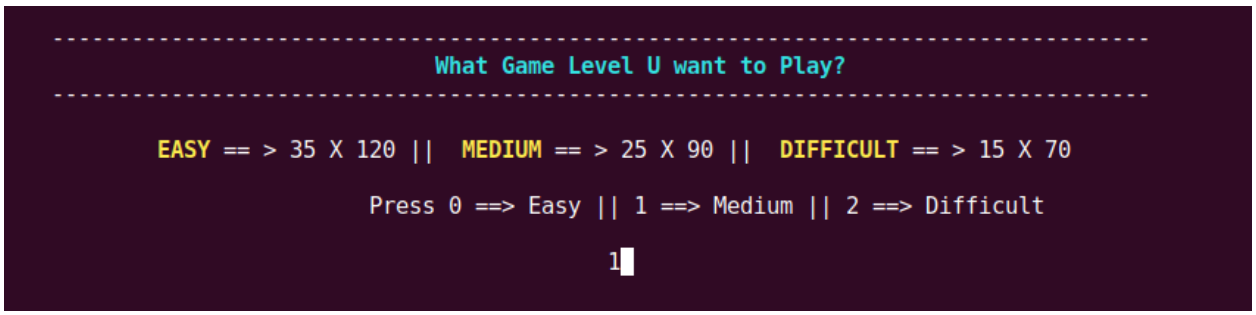


Fig.7.4 Game Level & user press ‘1’ for medium

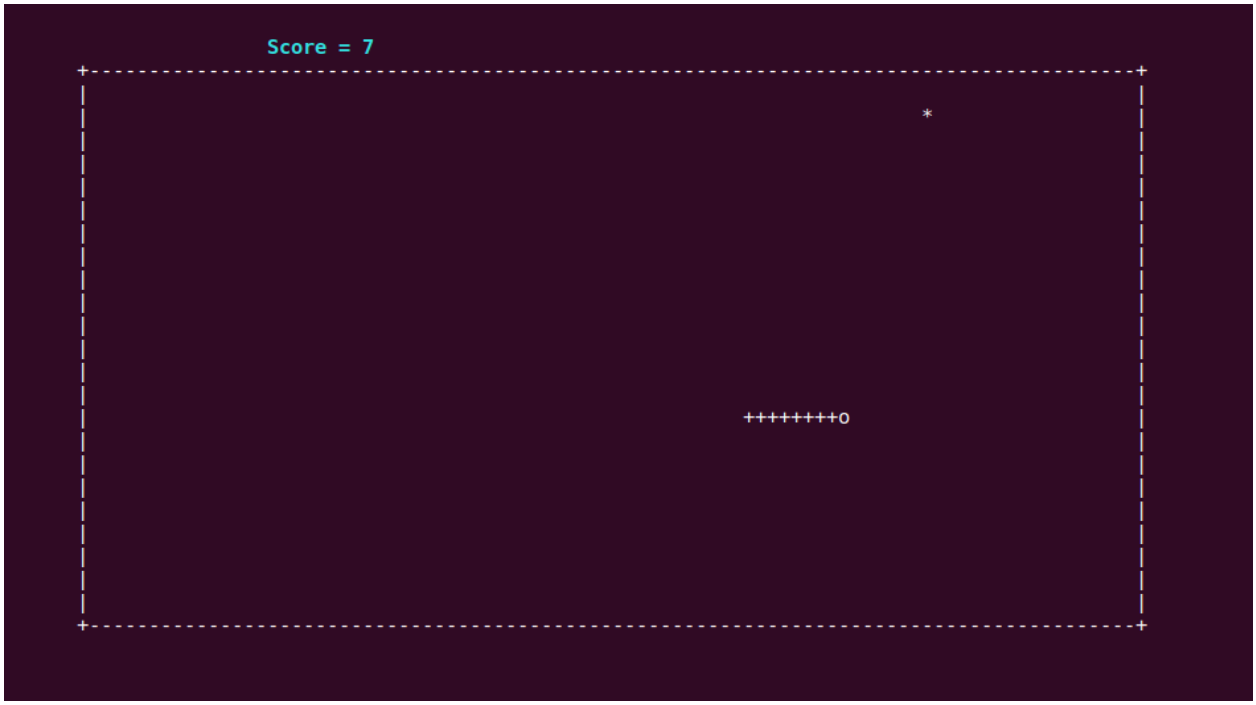


Fig.7.5 Medium level arena

Fig.7.6 Game Over

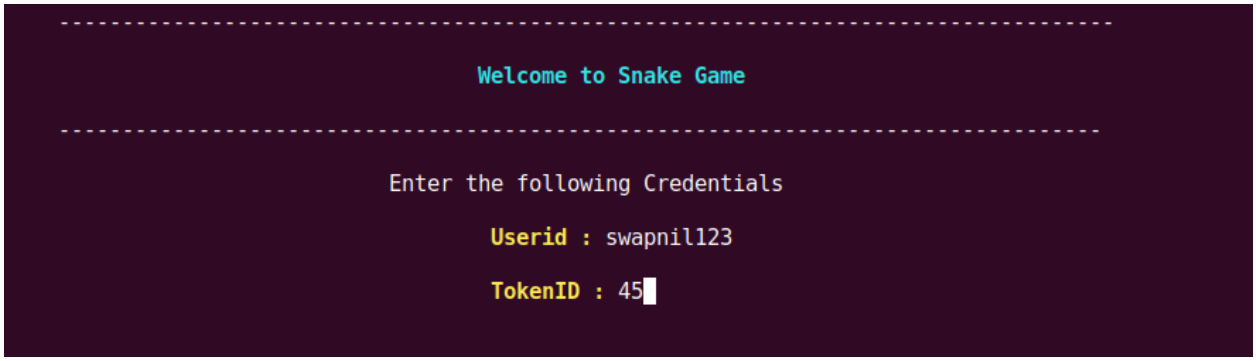


Fig.7.7 Sign Up

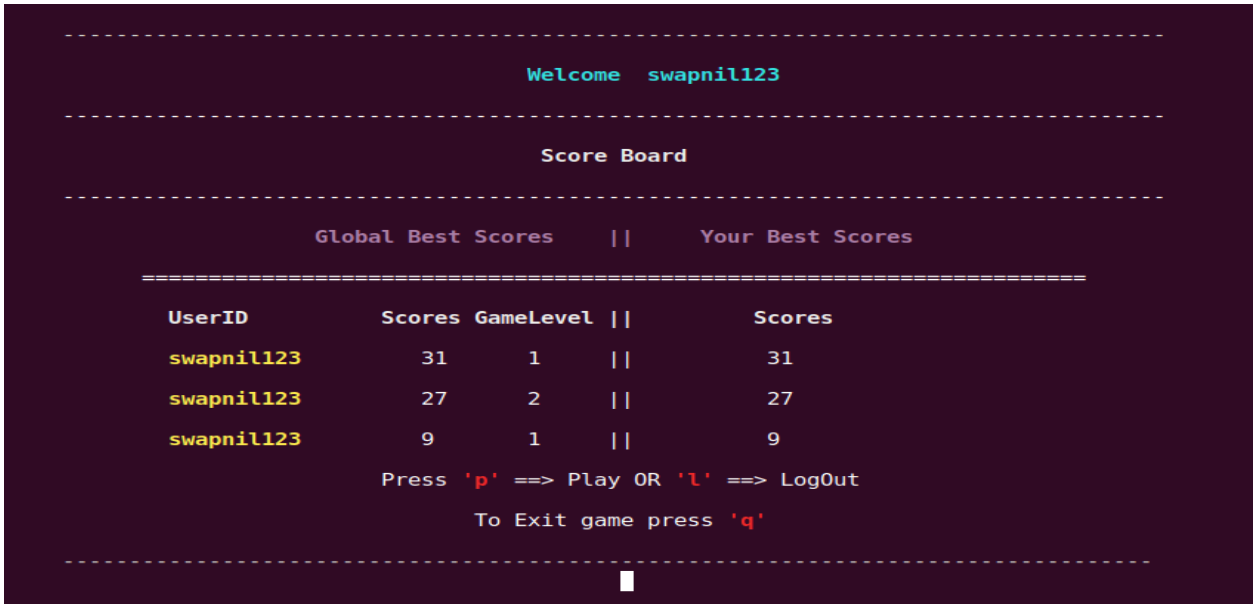
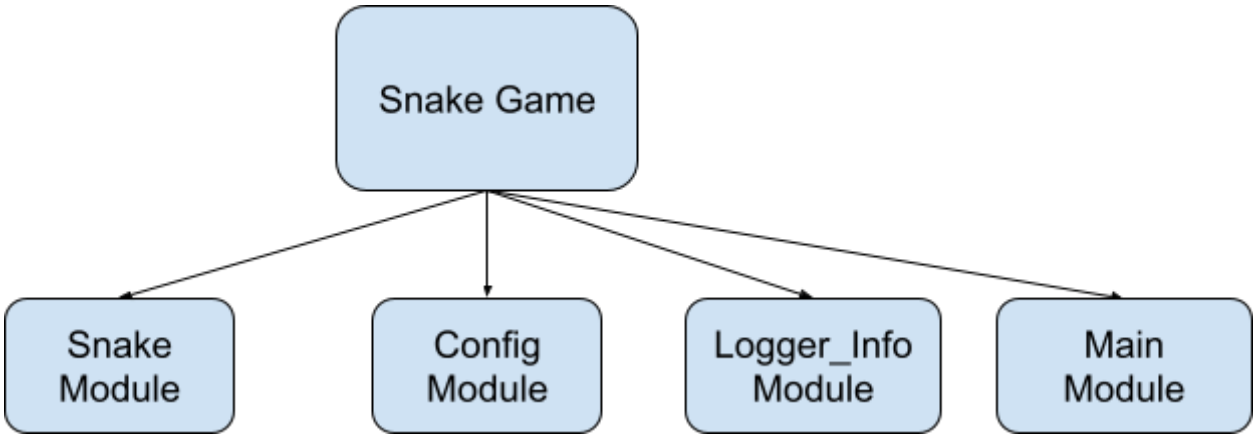


Fig.7.8 Main Menu

7.4. System Architecture

The Game development is being divided into three modules

(1) Snake
(2) Config
(3) Loggger_info



==> **Snake Module** will handle the initialization of parameters for the snake , movement of snake , Updating the score and life_count and saving whole data into the file.

It contains following API :

(1) **InitializeSnake() :** Initialize the boundary according to the game level the user has selected , Initialize life count , Place the snake at a random location in the arena.
(2) **MoveUp() :** Moves the snake in upward direction.
(3) **MoveDown() :** Moves the snake in downward direction.
(4) **MoveLeft() :** Moves the snake in the left direction.
(5) **MoveRight()** : Moves the snake in the right direction.
(6) **MoveStraight()** : Moves the snake in the same direction according to the last input.
(7) **Bend()** : Bends the snake at 90 degree according to the input.
(8) **UpdateScore()** : Updates the score each time when the snake eats the food.
(9) **UpdateLifeCount() :** Updates the LifeCount of the Snake.
(10) **StoreRecord() :** After the game gets over , the score is recorded in the file.
(11) **CreateFood() :** Create next Food each time , the snake eats.
(12) **ValidityCheck() :** Checks the condition whether the snake has touched the boundary or ate himself or not.
(13) **DeInitializeSnake() :** Closes the file opened for the snake , frees the memory , sets the default values.

==> **Config Module** will handle the reading the contents of the user , shows the welcome screen , Shows the score , Shows Game instructions and starts the game or over the game.

It contains following API :

(1) **InitializeConfig() :** Initialize the whole game by reading the config file.
(2) **NewUser() :** Stores the data of the new user in the file.

(3) **ExistingUser() :** finds the data of existing users from the file.
(4) **OwnScore() :** finds the old scores of game , the user has played up to top 5
(5) **GlobalScore() :** finds the top 5 score of all players who have played this game.
(6) **DeInitializeConfig() :** Closes the files opened for configuration , free memory ,set the default values.

==> **Logger_info Module** will handle the writing of the errors , information of files opened or closed or not and the debugging part in a separate file.

It contains following API :

(1) **InitializeLog()** : It will open the file for storing all the log's information , error and debugs.
(2) **LogInfo() :** Writes the Information of events happening in the game into the file.
(3) **LogDebug() :** Writes the debug information of the game into the file.
(4) **LogError() :** Writes the Error information of the game into the file.
(5) **NewGame() :** Writes the New game Message into the file.
(6) **DeInitializeLog() :** CLoses the files opened for the logger , frees memory and sets the default values.

==> **Main Module** will handle the sequence of calling the function for execution and prints the messages on the prompt.

It contains following API :

(1) **WelcomeScreen() :** Shows a welcome message on the screen and asks the user whether he/she is a new player or not.
(2) **ShowScore() :** Shows the score of the global player and the user's score.
(3) **GameInstructions() :** Shows the Game Instructions of how to play this game.
(4) **ShowLifeCount() :** Shows the life count of the snake.
(5) **StartGame() :** Starts the game by initializing the level and the boundary selected by the user.
(6) **GameOver() :** prints the message and checks whether the user has beaten its own score or not.

7.5. Functional Requirement

Req. ID	Requirement description
SR_1	Program should handle the invalid input during the registration process & during the game mode.
SR_2	Program should not get crashed , if the snake size gets larger than the boundary area.
SR_3	Instant response is required , when the user presses any keys.

Table 7.1 Functional Requirement

Conclusion

In conclusion this training is amazing to start a career in embedded systems. We have gained very deep knowledge of c programming language including all basic concepts of c language, doxygen style commenting to generate html pages, memory organization, valgrind tool, makefile pointers, structured programming & gdb. We also learned about gitlab in which all the basic commands & git structure was covered. We also learned linux operating systems in which we covered all the basics as well as all user & network administration commands and also learned about different services like samba,nfs etc. Finally we went through the training of Data Structures in which we learnt about link lists and it's different operations and modes. We also performed binary tree and binary search tree operations. So this industrial training is very good to complete educational requirements in 8th semester and to start an industrial career.

References

https://cs-fundamentals.com/c-programming/memory-layout-of-c-program-code-data-segments

https://www.toolsqa.com/git/what-is-git/

https://www.javatpoint.com/linux-system-admin-commands

http://www.linuxandubuntu.com/home/what-is-samba-server-and-how-to-setup-samba-server-in-ubuntu-linux#:~:text=Samba%20is%20an%20open%2Dsource,Message%20Block%20(SMB)%20protocol

https://bash.cyberciti.biz/guide/Service_command#:~:text=The%20service%20command%20is%20used%20to%20run%20a%20System%20V%20init%20script.&text=d%20directory%20and%20service%20command,%2C%20stop%2C%20and%20restart%20commands

https://www.tecmint.com/how-to-setup-nfs-server-in-linux/#:~:text=NFS%20(Network%20File%20System)%20is,locally%20on%20the%20same%20system