Mawlana Bhashani Science and Technology University
Department of Information and Communication Technology
3rd Year 1st Semester, B.Sc. (Engg.) in ICT
Course Title: Operating System
Course Code: ICT-3110

Name: Ali Ashadullah Arif **ID: IT-18031**

Table: Lab Report

Lab Report : 01

Name of the lab report : Installing Linux

Objective: In this lab we just see how to install linux os in our device.

Installing Linux

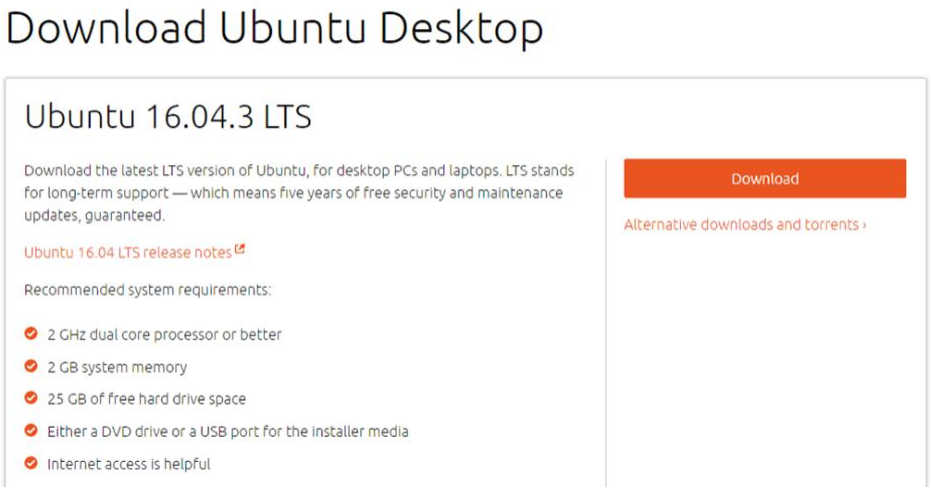
Let's look the various methods we can use to install Ubuntu.

Installing Linux using USB stick

This is one of the easiest methods of installing Ubuntu or any distribution on your computer.

Follow the steps.

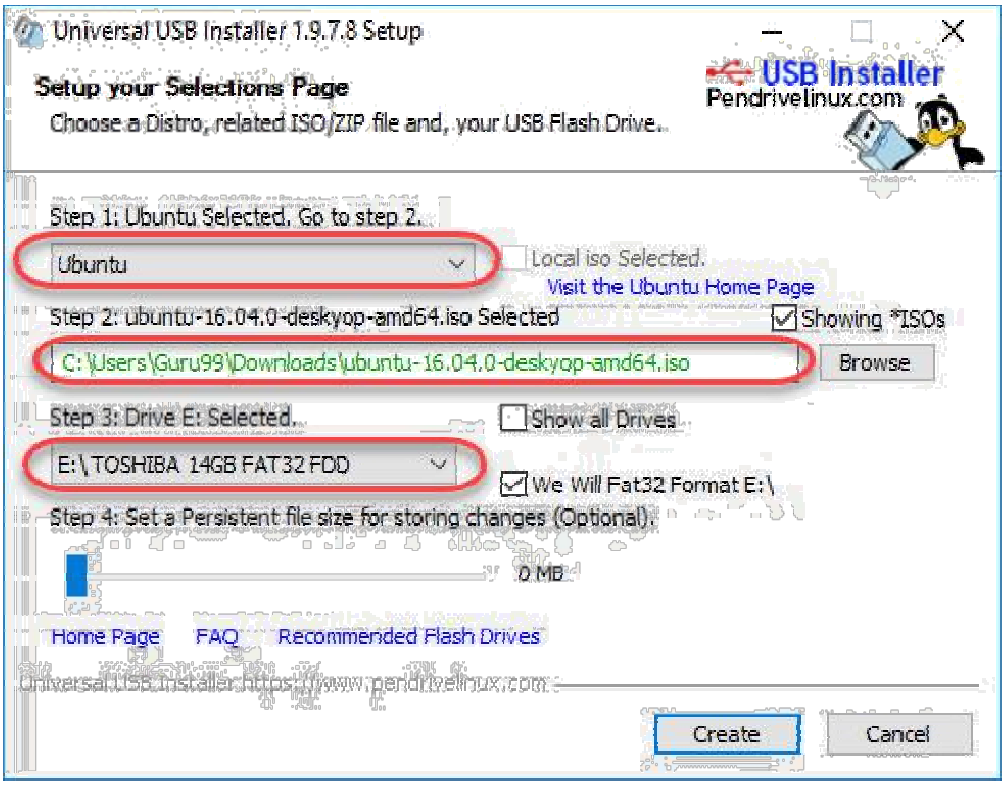
Step (1) Download the .iso or the OS files on your computer from this link.



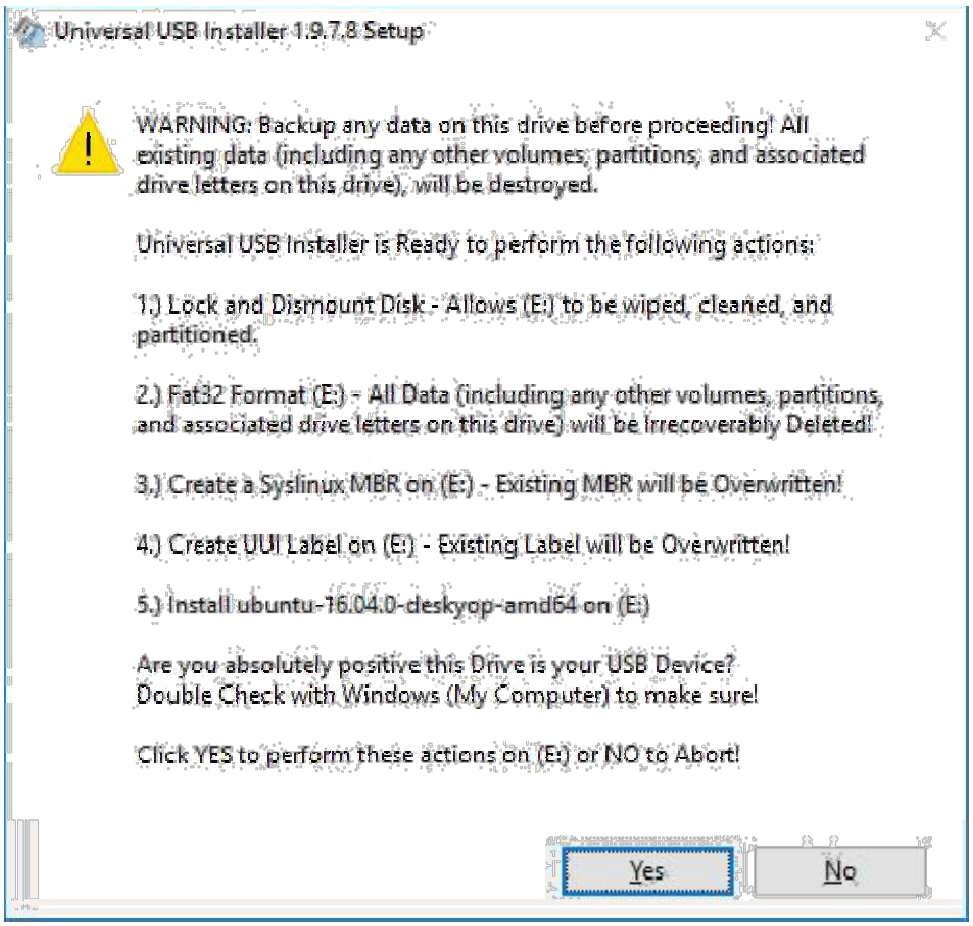
Step (2) Download free software like 'Universal USB installer to make a bootable USB stick.

Step 3) Select an Ubuntu Distribution form the dropdown to put on your USB Select your Ubuntu iso file download in step 1.

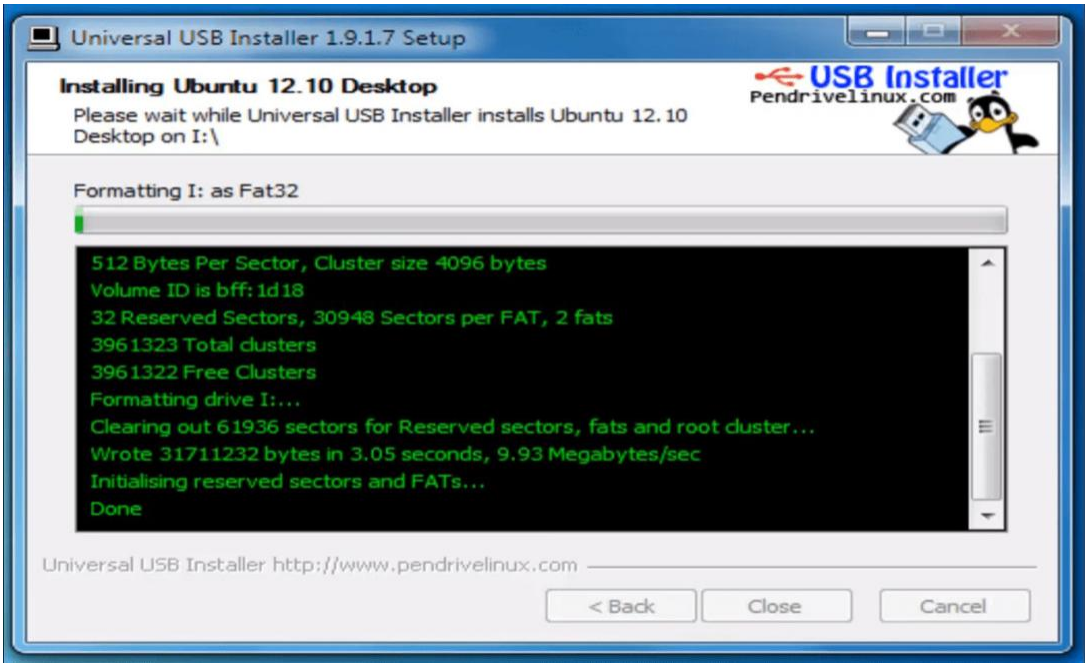
Select the drive letter of USB to install Ubuntu and Press create button



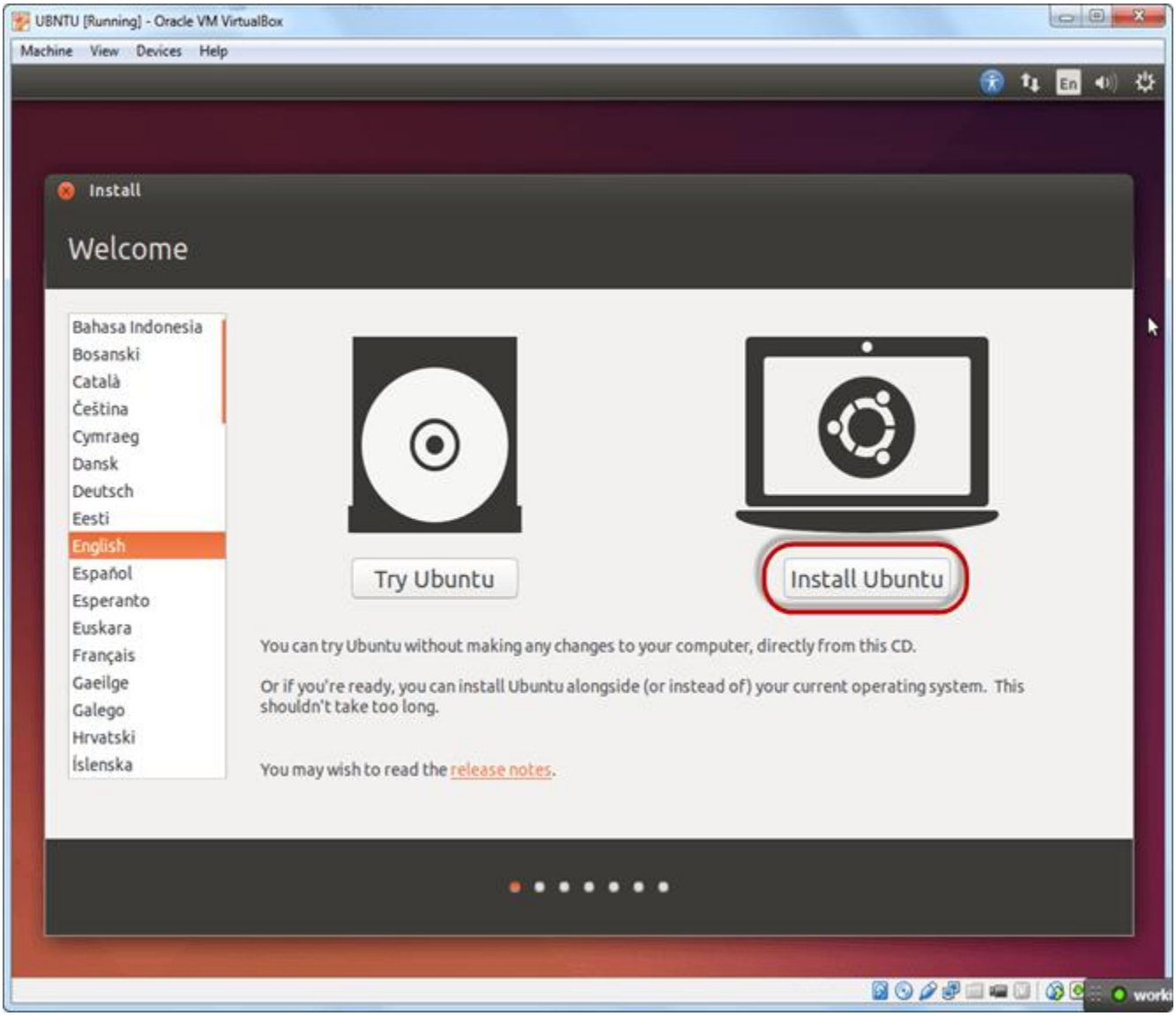
Step 4) Click YES to Install Ubuntu in USB.



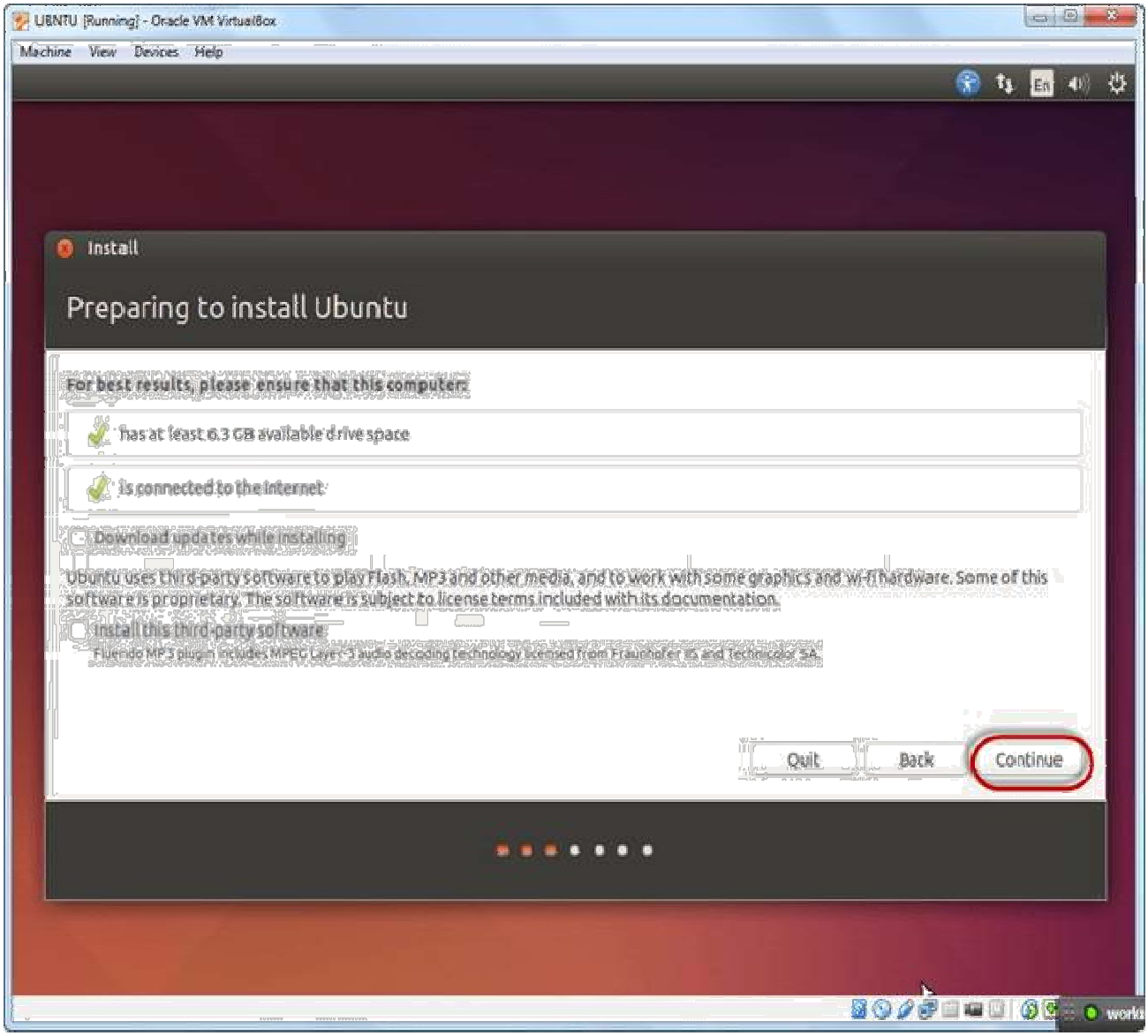
Step (5) After everything has been installed and configured, a small window will appear Congratulations! You now have Ubuntu on a USB stick, bootable and ready to go.



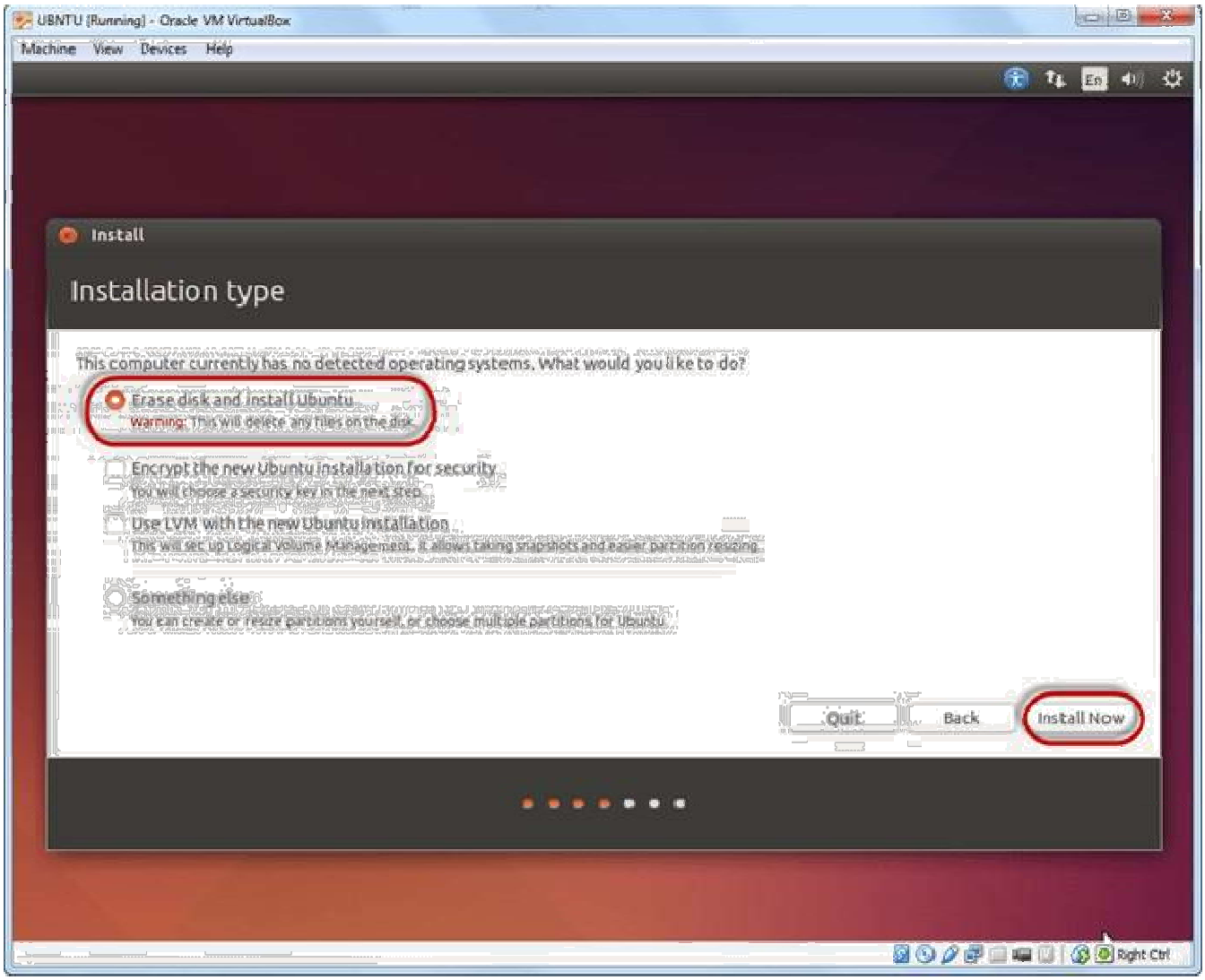
Step-6 You have an option to Run Ubuntu WITHOUT installing. In this tutorial will install Ubuntu



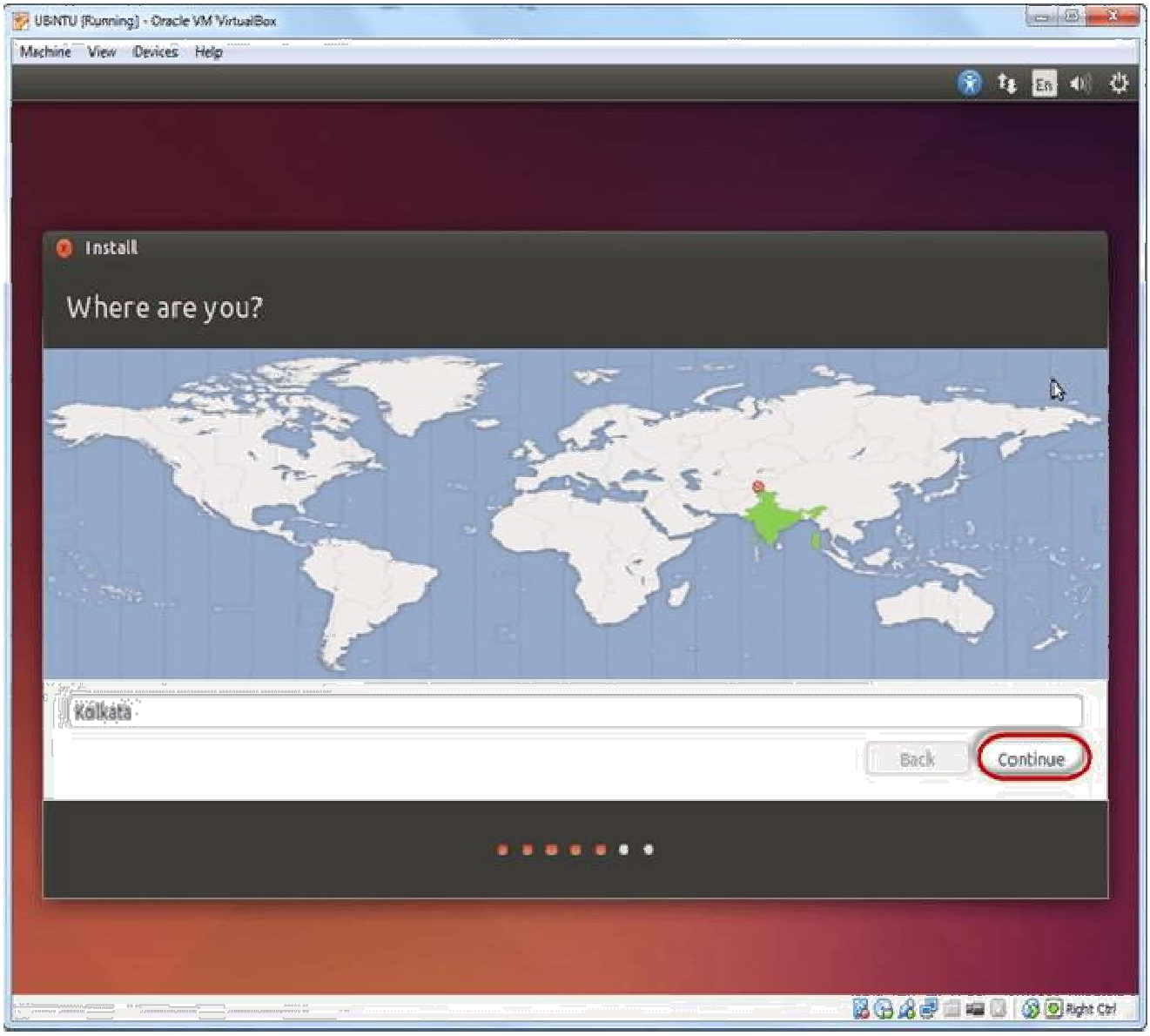
Step-7) Click continue.



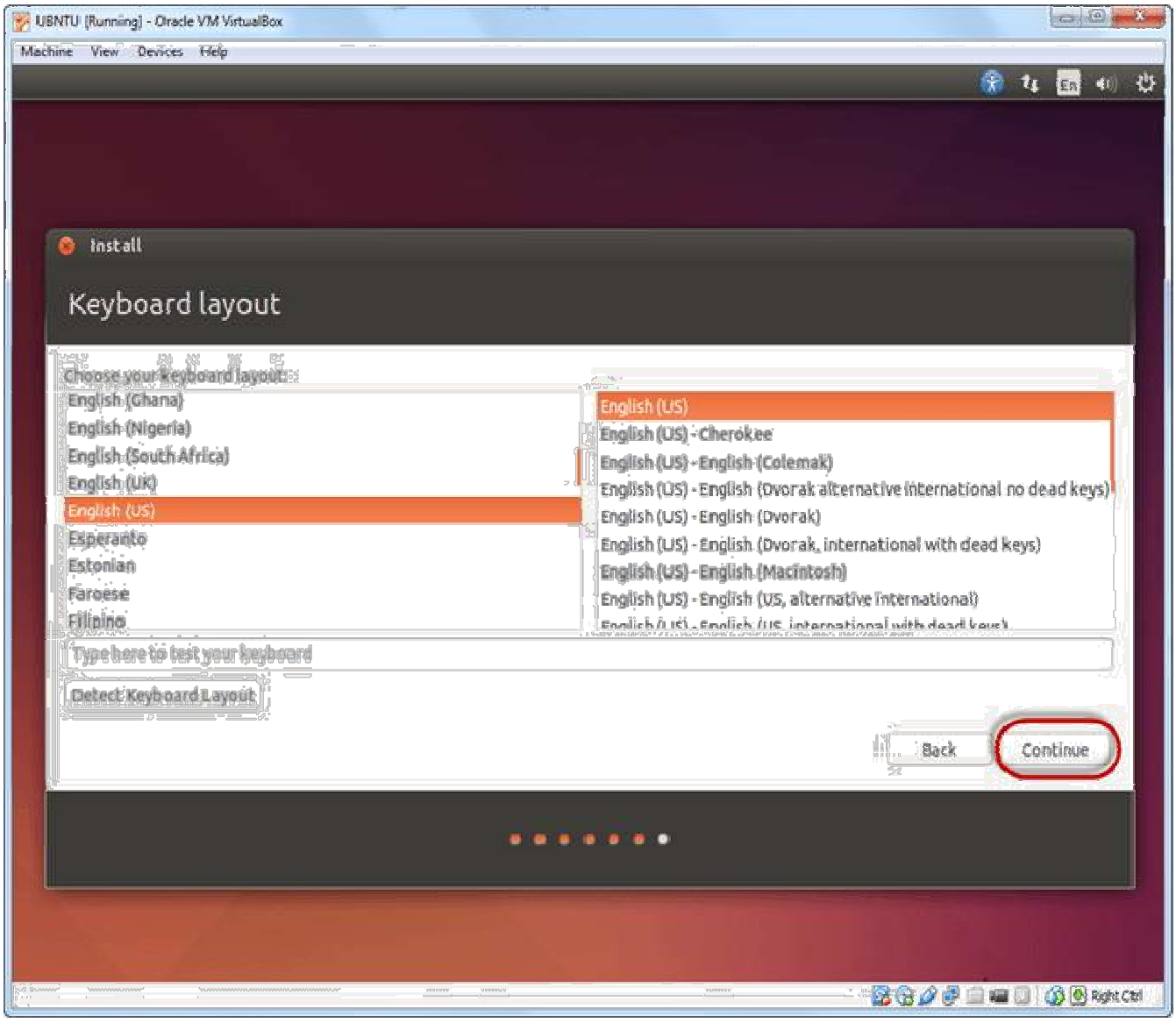
Step (8) Select option to erase the disk and install Ubuntu and click on install now. This option installs Ubuntu into our virtual hard drive which is we made earlier. It will not harm your PC or Windows installation



Step (9) Select your location for setting up time zone, and click on continue



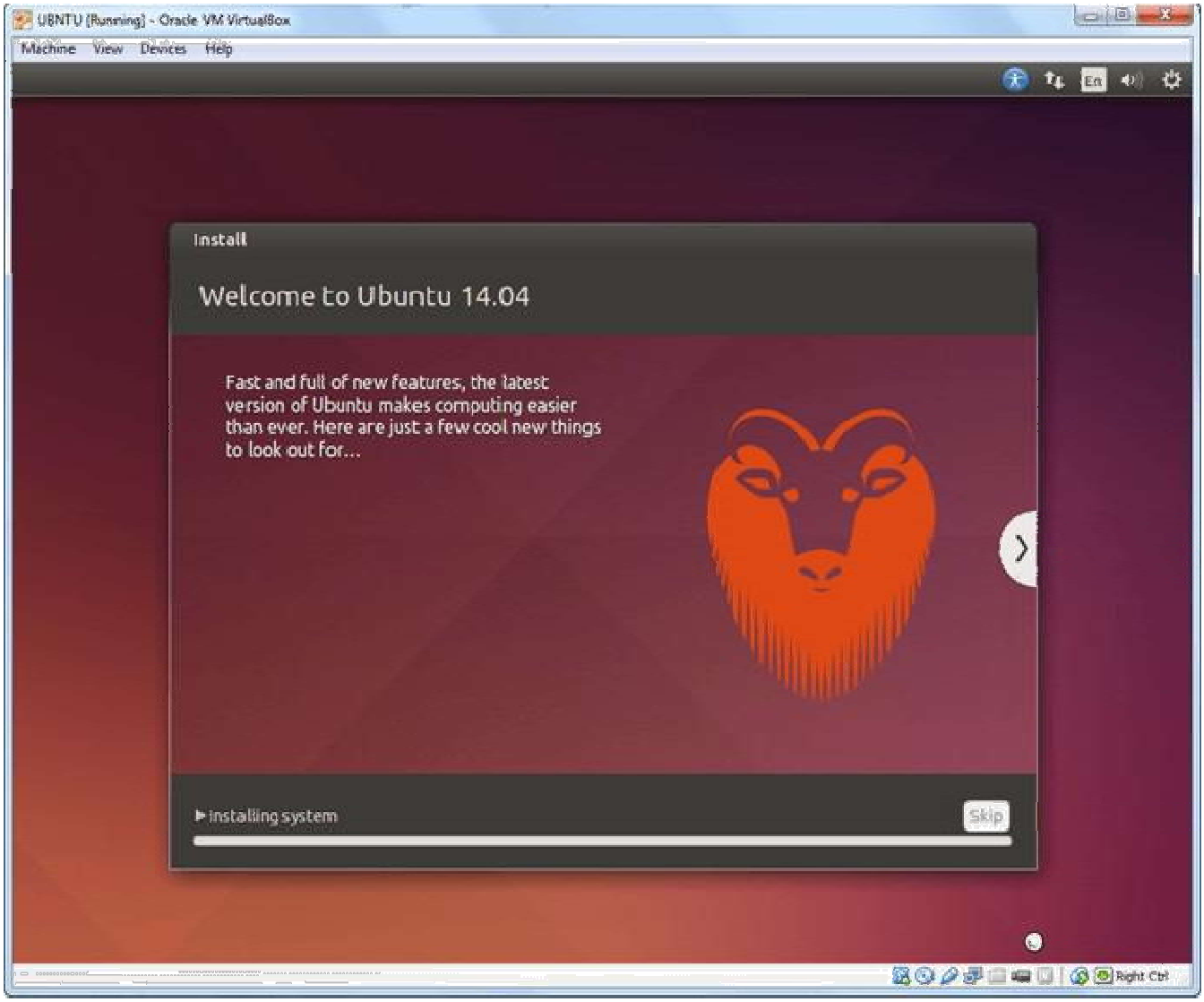
Step (10) Select your keyboard layout, by default English (US) is selected but if you want to change then, you can select in the list. And click on continue



Step (11) Select your username and password for your Ubuntu admin account. This information has been needed for installing any software package into Ubuntu and also for login to your OS.

Fill up your details and tick on login automatically to ignore login attempt and click on continue

Step (12) Installation process starts. May take up to 30 minutes. Please wait until installation process completes.



Step (13) After finishing the installation, you will see Ubuntu Desktop.

Conclusion: In this lab, we learnt about how to install linux. This is an another operating system. With Linux, we do so many different types of task with terminal and with specific commands. This operating system is different, but easy and useful.

Lab Report No: 03

Name of the Lab Report: Basic Command of Linux Operating System

1. **What is Linux command?**

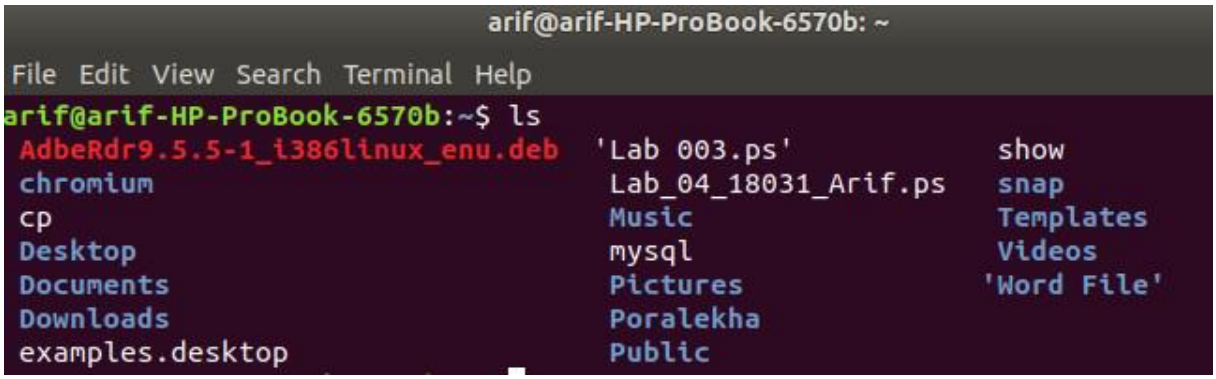
Answer: Linux is a Unix-Like operating system. All the Linux/Unix commands are run in the terminal provided by the Linux system. This terminal is just like command prompt of Windows OS. Linux/Unix commands are case-sensitive, terminal can be used to accomplish all Administrative tasks. This includes package installation, file manipulation, and user management. Linux terminal is user-interactive. The terminal outputs the results of commands which are specified by the user itself.

2. **Describe the operation of Linux basic command(screenshot)**

Answer : The operation of basic Linux command is given below :

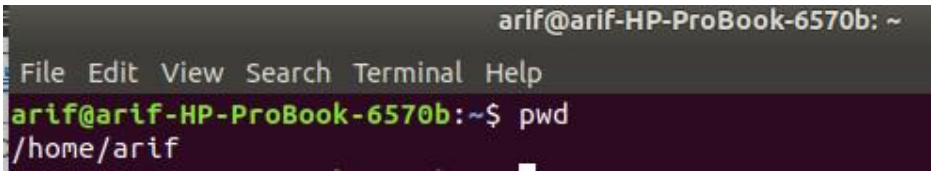
1) ls : ls command is used for listing contents of a directory. It works as dir command.

Example :



2) pwd : pwd command displays the name of current/working directory as below.

Example:


arif@arif-HP-ProBook-6570b: ~
File Edit View Search Terminal Help
arif@arif-HP-ProBook-6570b:~$ pwd
/home/arif

3) chmod: chmod command is used to change/update file access permissions like this .

Example:

```
Usage: chmod [OPTION]... MODE[,MODE]... FILE...
  or:  chmod [OPTION]... OCTAL-MODE FILE...
  or:  chmod [OPTION]... --reference=RFILE FILE...
Change the mode of each FILE to MODE.
With --reference, change the mode of each FILE to that of RFILE.

  -c, --changes          like verbose but report only when a change is made
  -f, --silent, --quiet  suppress most error messages
  -v, --verbose          output a diagnostic for every file processed
      --no-preserve-root  do not treat '/' specially (the default)
      --preserve-root    fail to operate recursively on '/'
      --reference=RFILE  use RFILE's mode instead of MODE values
  -R, --recursive        change files and directories recursively
      --help     display this help and exit
      --version  output version information and exit

Each MODE is of the form '[ugoa]*([-+=]([rwxXst]*|[ugo]))+|[-+=][0-7]+'.

GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
Full documentation at: <http://www.gnu.org/software/coreutils/chmod>
or available locally via: info '(coreutils) chmod invocation'
```

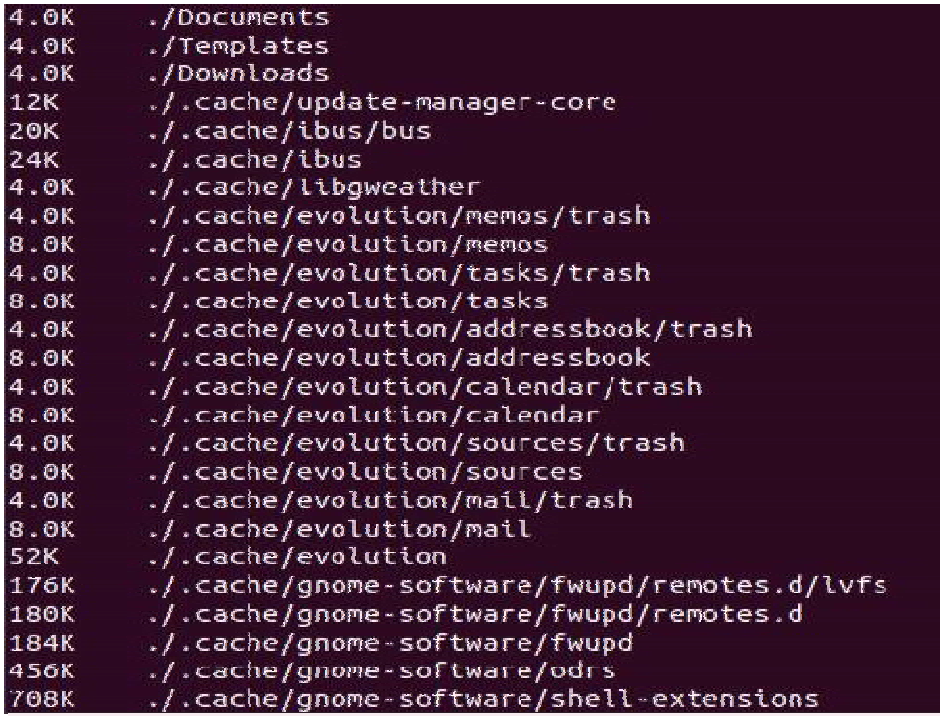
4) df: df command is used to show file system disk space usage as follows.

Example:

```
Filesystem     1K-blocks    Used Available Use% Mounted on
sysfs                  0       0         0    - /sys
proc                   0       0         0    - /proc
udev              479464       0    479464   0% /dev
devpts                 0       0         0    - /dev/pts
tmpfs             100668    1544     99124   2% /run
/dev/sda1       30830500 5998968  23242388  21% /
securityfs             0       0         0    - /sys/kernel/security
tmpfs             503336       0    503336   0% /dev/shm
tmpfs               5120       4      5116   1% /run/lock
tmpfs             503336       0    503336   0% /sys/fs/cgroup
cgroup                 0       0         0    - /sys/fs/cgroup/unified
cgroup                 0       0         0    - /sys/fs/cgroup/systemd
pstore                 0       0         0    - /sys/fs/pstore
cgroup                 0       0         0    - /sys/fs/cgroup/net_cls,net_prio
cgroup                 0       0         0    - /sys/fs/cgroup/hugetlb
```

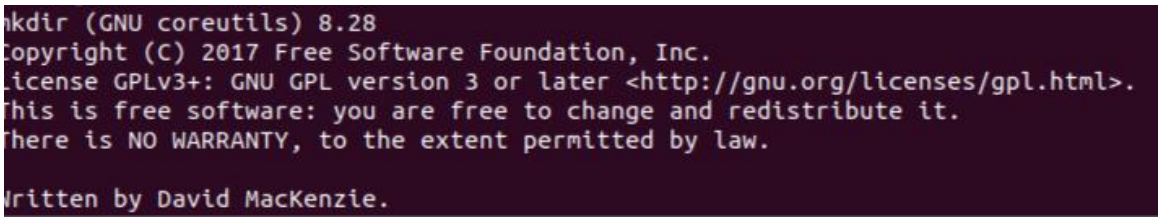
5) du: du command is used to show disk space usage of files present in a directory as well as its sub – directories .

Example :



6) mkdir: mkdir command is used to create single or more directories, if they do not already exist (this can be overridden with the –p option).

Example:



7) passwd : passwd command is used to create or update passwords for user accounts, it can also change the account or associated password validity period.

Example:

8) rm : rm command is used to remove files or

directories. Example:

```
Usage: rm [OPTION]... [FILE]...
Remove (unlink) the FILE(s).

  -f, --force           ignore nonexistent files and arguments, never prompt
  -i                    prompt before every removal
  -I                    prompt once before removing more than three files, or
                          when removing recursively; less intrusive than -i,
                          while still giving protection against most mistakes
      --interactive[=WHEN]  prompt according to WHEN: never, once (-I), or
                          always (-i); without WHEN, prompt always
      --one-file-system  when removing a hierarchy recursively, skip any
                          directory that is on a file system different from
                          that of the corresponding command line argument
      --no-preserve-root  do not treat '/' specially
      --preserve-root   do not remove '/' (default)
  -r, -R, --recursive   remove directories and their contents recursively
  -d, --dir             remove empty directories
  -v, --verbose         explain what is being done
      --help     display this help and exit
      --version  output version information and exit
```

9) ln : ln command is used to create a soft link between files using the –s flag .

Example :

```
ln (GNU coreutils) 8.28
Copyright (C) 2017 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Written by Mike Parker and David MacKenzie.
```

10) tar : tar command is a most powerful utility for archiving files in Linux.

Example :

```
tar (GNU tar) 1.29
Copyright (C) 2015 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

11) cd : cd stands for change directory and it does the same as it name stands for.

Example :

Conclusion: In this Lab, we learnt about the basic command of linux. We are apply this command in linux to do many types of task, basically huge types of operations. So, this lab report is very helpful for starting this operating system.

Lab No : 03

Name of the lab Report: Threads on Operating System

Objective: In this lab ,we can learn about Thread, Threads type and Threads Implementation and how it works in operating system.

Threads on Operating System

Q1. What is Thread?

Answer:

Thread: A thread is the unit of execution within a process. A process can have anywhere from just one thread to many threads.

A thread is a basic unit of CPU utilization that shares with other threads belonging to the same process it's code section,data section and other operating system resources such as open files and signals.

Q2. Types of Threads?

Answer:

Threads are divided into parts. They are given below –

1. User Threads.
2. Kernel Threads

User Threads: This threads support above the kernel and managed without kernel support. This type of thread library contains code for creating and destroying threads, for passing message and data between threads, for scheduling thread execution and for saving and restoring thread contexts. The main application in this threads starts with a single thread.

Advantages:

1. Context switch required no hardware supports.

2. Context switch time is less.
3. Thread switching does not require Kernel mode .
4. User thread can run on any operating system.
5. Implement of user threads is easy.

Disadvantages:

1. In a typical operating system, most system calls are blocking.
2. Multi-threaded application cannot take advantage of multiprocessing

Kernel Threads: Kernel-level threads are handled by the operating system directly and the thread management is done by the kernel. The context information for the process as well as the process threads is all managed by the kernel. Because of this, kernel-level threads are slower than user-level threads.

Advantages:

1. Kernel can simultaneously schedule multiple threads .
2. If one thread in a process is blocked, the Kernel can schedule another thread of the same process.
3. Kernel routines themselves can be multithreaded.

Disadvantages:

1. Hardware supports is needed.
2. Implementation of kernel thread is complicated.

Q3. How to implement a Thread?

Answer:

There are two ways of implementing a thread :

1. Threads implementation in user space
2. Threads implementation in kernel

Now describe briefly about the above two ways of implementing a thread.

1. Threads implementation in user space :

In this model of implementation, the threads package entirely in user space, the kernel has no idea about it. A user thread package can be executed on an operating system that does not support threads and this is the main advantage of this implementation model. All of these implementations have the same general structure as illustrated in the figure given below.

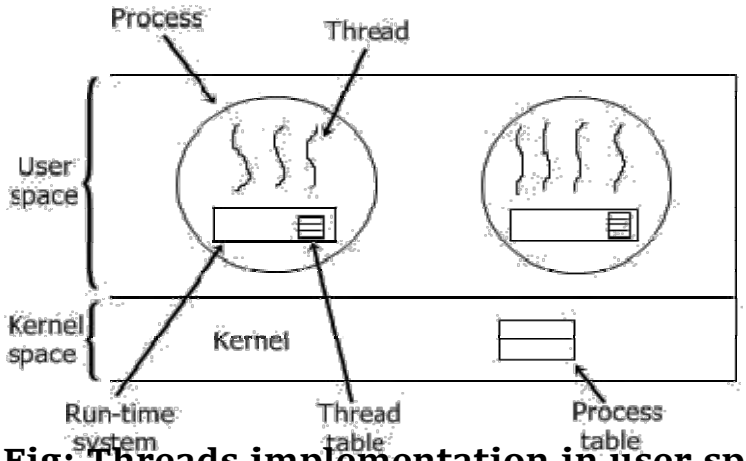


Fig: Threads implementation in user space

2. Threads implementation in kernel:

In this method of implementation, the threads package entirely in the kernel, no any run-time system is need in each as illustrated in the figure given below.

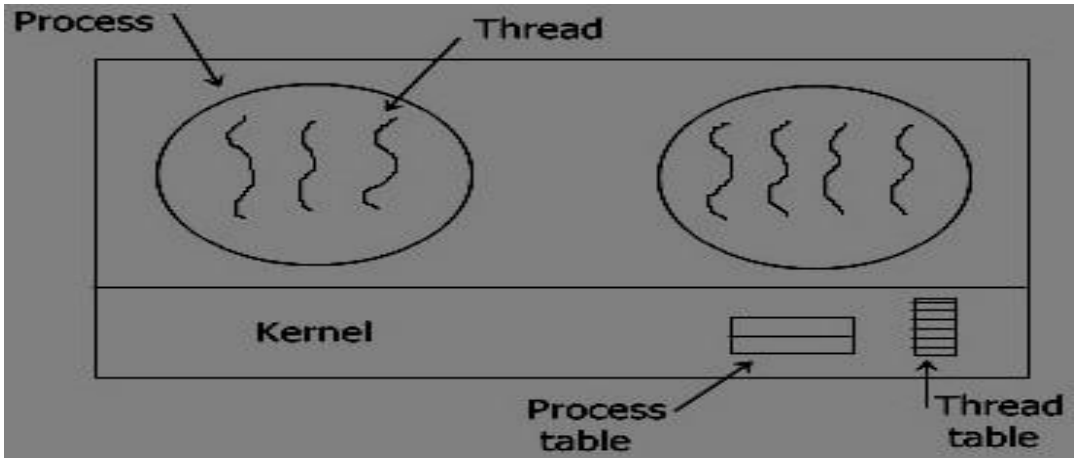


Fig: Threads implementation in kernel

In this, there is no any thread table in each process. But to keep track of all the threads in the system, the kernel has the thread table .Whenever a thread wants to create a new thread or destroy an existing thread, then it makes a kernel call, which does the creation or destruction just by updating the kernel thread table .The thread table of the kernel holds each registers, state, and some other useful information of the thread. In this method of implementation model, the threads package completely in the kernel. There is no need for any run-time system. To maintain the record of all threads in the system a kernel has a thread table.

Conclusion: By doing this lab report, I learnt about the basic concept of thread, thread types, how it can be implemented and also how it works in operating system by using kernel. Mainly, threads are used multiple application running at a same time period in a processor. The main benefit of using thread is that we can do multiple task by dividing a process into multiple threads.

Lab Report No: 04

Name of the lab report: File operation and permission.

Objective: The file system is the most obvious aspect of any OS. This provides users the method for storage and access to data as well as programs of the operating system. In this report paper we know a lot about File operating process on Linux OS.

Q1: What is File Operation and File Permission in Linux Operating System?

Answer:

File Operation: Just likely in Linux operating system, everything is organized in the form of files and directories. By setting permissions on files and directories, one can make sure that only authorized users are allowed to access a specific data. Each file in Linux is owned by a user and group. The user is the one who creates the file and group is the one to which the user belongs to.

File Permission: File permissions consist of three permissions that you can apply to files and directories. In this section, you'll learn how the system works and how to modify these permissions. Before doing this, let's have a look at how to read the current permissions. The best method to do so is by using ls-l which will show you a list of all files and directories in the current directory.

1. The first column shows the file permissions.
2. The third column shows the user owner of the file.
3. The fourth column shows the group owner of the file.

Q2. Implementation of File Operation and File Permission.

Answer:

Numerous on-disk and in-memory configurations and structures are being used for implementing a file system and the file system but applying some general principles. Here they are portrayed below:

1. A boot control block usually contains the information required by the system for booting an operating system from that volume. When the disks do not contain any operating system, this block can be treated as empty. This is typically the first chunk of a volume. In UFS, this is termed as the boot block; in NTFS, it is the partition boot sector.

2. A volume control block holds volume or the partition details, such as the number of blocks in the partition, size of the blocks or chunks, free-block count along with free-block pointers. In UFS, it is termed as superblock; in NTFS, it is stored in the master file table.

3. A directory structure per file system is required for organizing the files. In UFS, it held the file names and associated 'inode' numbers. In NTFS, it gets stored in the master file table.

4. The FCB contains many details regarding any file which includes file permissions, ownership; the size of file and location of data blocks. In UFS, it is called the inode. In NTFS, this information gets stored within the master file table that uses a relational database (RDBM) structure, using a row per file.

Permission Groups:

Each file and directory has three user based permission groups:

Owner: The Owner permissions apply only the owner of the file or directory, they will not impact the actions of other users.

Group: The Group permissions apply only to the group that has been assigned to the file or directory, they will not effect the actions of other users.

All user: The All Users permissions apply to all other users on the system, this is the permission group that you want to watch the most.

Permission Types:

Each file or directory has three basic permission types:

Read: The Read permission refers to a user's capability to read the contents of the file.

Write: The Write permissions refer to a user's capability to write or modify a file or directory.

Execute: The Execute permission affects a user's capability to execute a file or view the contents of a directory.

Conclusion: File Management, Some operating systems offer access to hardware through file management. Linux, which is part of Android OS is an example. So in Android, you can't make anything useful without it. It's also possible to make an operating system without files. It's valid to treat a disk as if it is just a big block of memory.

Reference: Zahid Hasan (IT-18017)

Lab Report No: 05

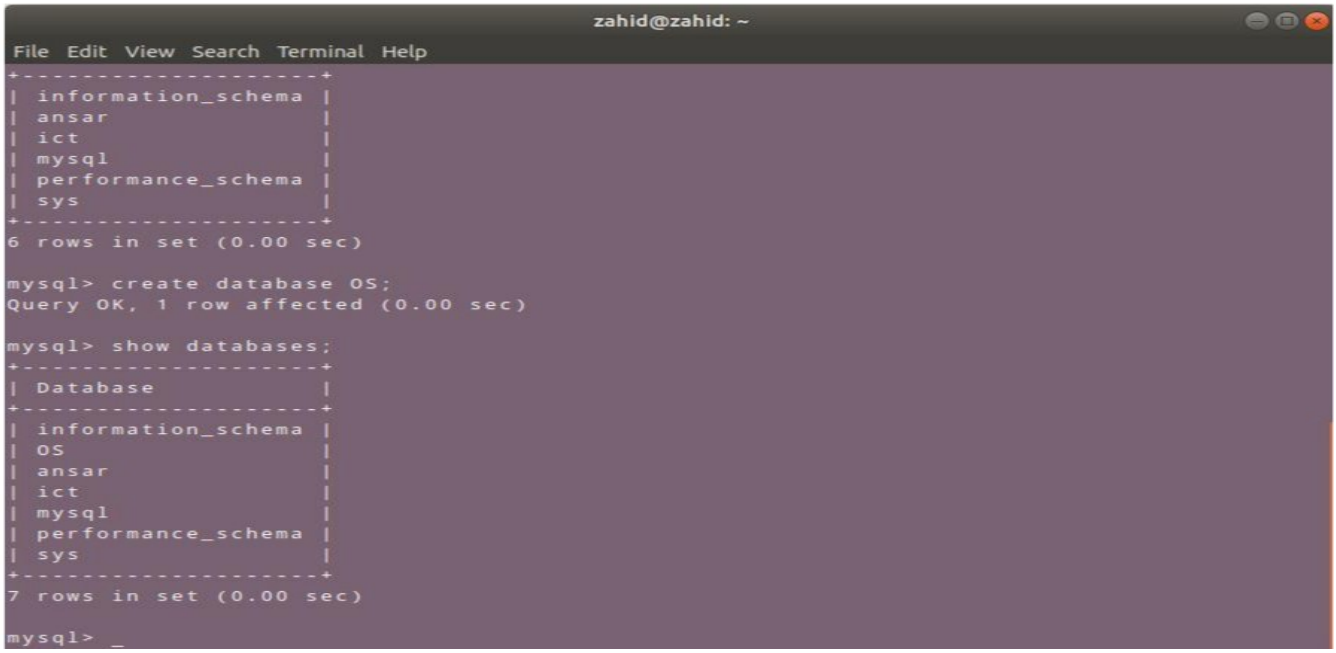
Lab Report Name: Connecting a database (MySQL) with linux

1. Install MySQL on Ubuntu

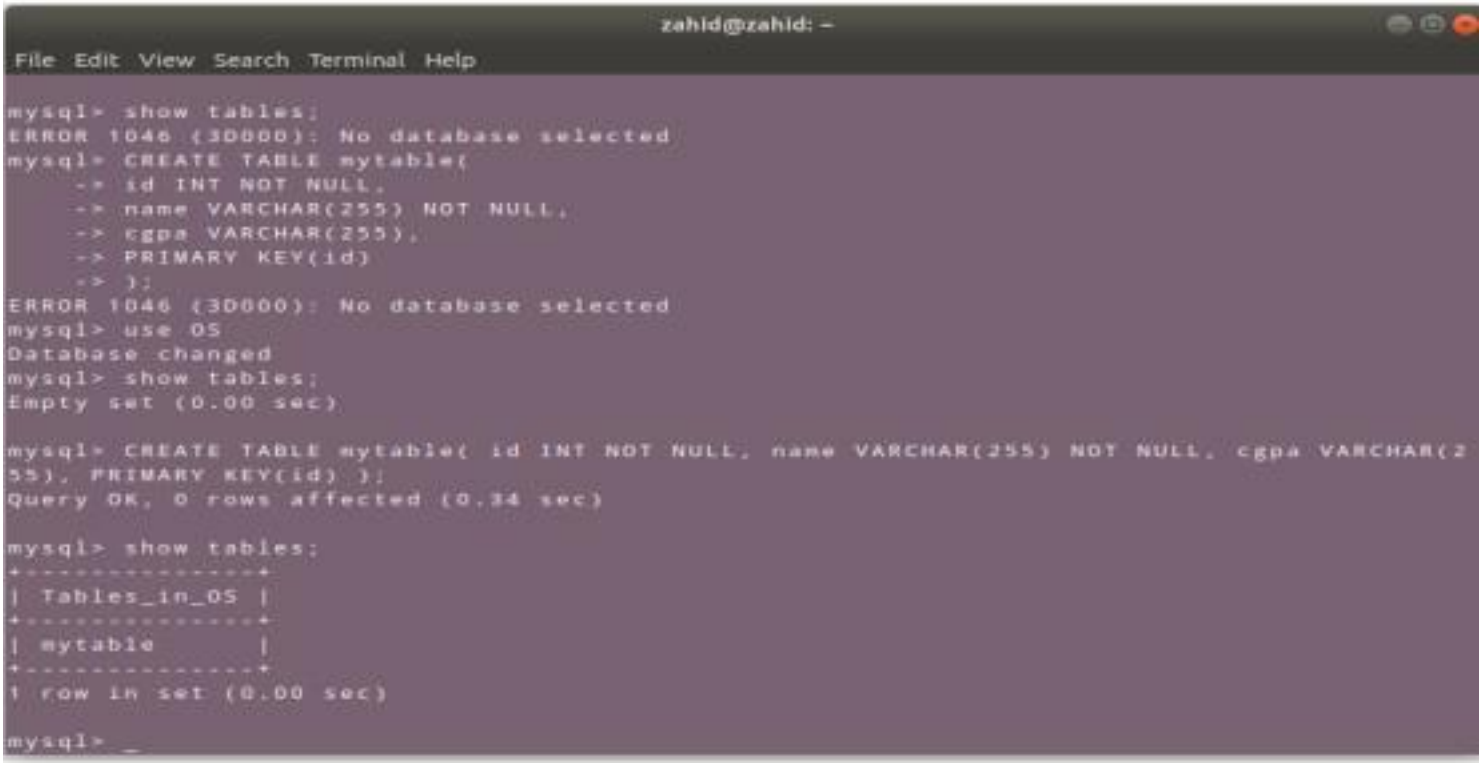
sudo apt-get install **mysql-server**

2. Log into mysql by linux ubuntu.

3. show and create database :



4. Show and create table :



5. Insert data into table:

```
nysql> INSERT INTO mytable values(18017,'Zahid Hasan','ICT');
Query OK, 1 row affected (0.09 sec)

nysql> SELECT* FROM mytable;
+-------+------------+------+
| id    | name       | cgpa |
+-------+------------+------+
| 18017 | Zahid Hasan | ICT  |
+-------+------------+------+
1 row in set (0.00 sec)

nysql>
```

6. Describe table:

```
ysql> DESCRIBE STUDENTS;
ERROR 1146 (42S02): Table 'OS.STUDENTS' doesn't exist
ysql> DESCRIBE mytable;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| id    | int(11)      | NO   | PRI | NULL    |       |
| name  | varchar(255) | NO   |     | NULL    |       |
| cgpa  | varchar(255) | YES  |     | NULL    |       |
+-------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

ysql>
```

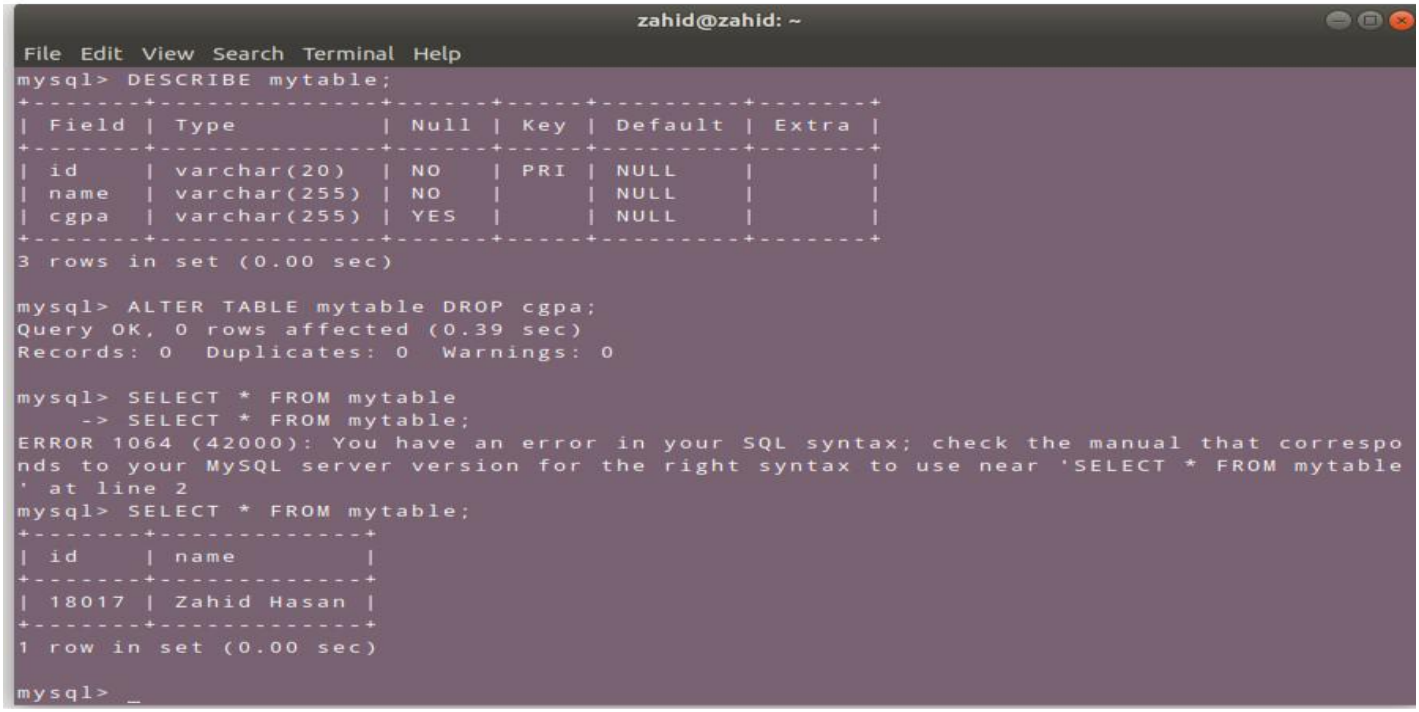
7. alter table :

```
mysql> ALTER TABLE student MODIFY id VARCHAR(20);
ERROR 1146 (42S02): Table 'OS.student' doesn't exist
mysql> ALTER TABLE mytable MODIFY id VARCHAR(20);
Query OK, 1 row affected (0.48 sec)
Records: 1  Duplicates: 0  Warnings: 0

mysql> DESCRIBE mytable;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| id    | varchar(20)  | NO   | PRI | NULL    |       |
| name  | varchar(255) | NO   |     | NULL    |       |
| cgpa  | varchar(255) | YES  |     | NULL    |       |
+-------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql>
```

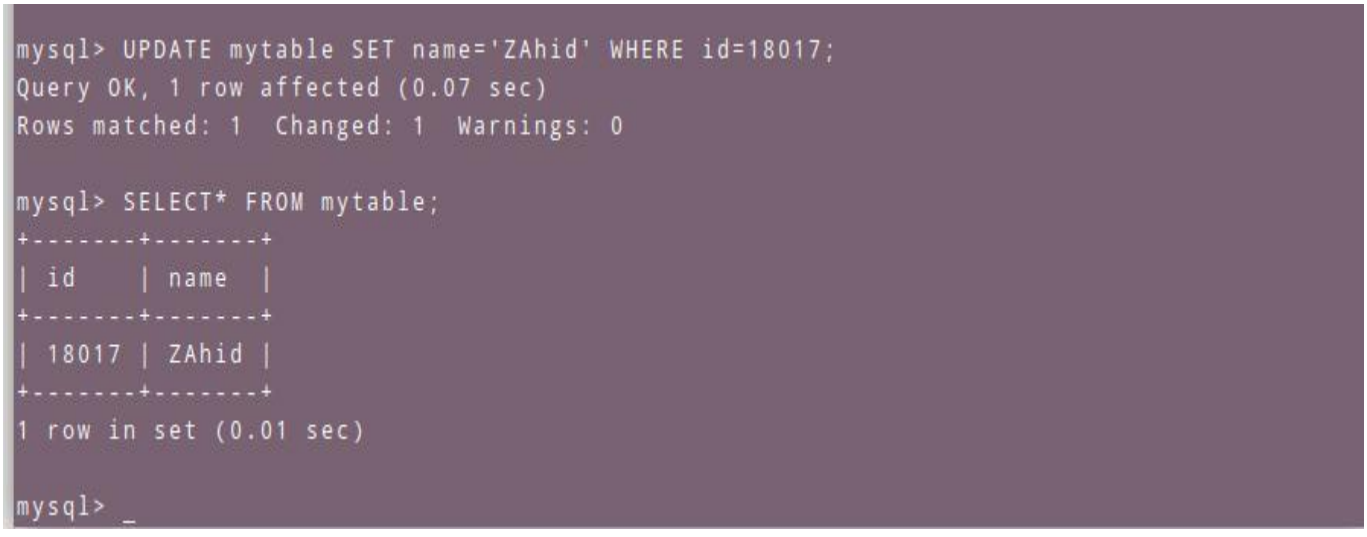
8. modify table:

```
mysql> ALTER TABLE student MODIFY id VARCHAR(20);
ERROR 1146 (42S02): Table 'OS.student' doesn't exist
mysql> ALTER TABLE mytable MODIFY id VARCHAR(20);
Query OK, 1 row affected (0.48 sec)
Records: 1  Duplicates: 0  Warnings: 0

mysql> DESCRIBE mytable;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| id    | varchar(20)  | NO   | PRI | NULL    |       |
| name  | varchar(255) | NO   |     | NULL    |       |
| cgpa  | varchar(255) | YES  |     | NULL    |       |
+-------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql>
```

9. drop data from table:



10.update data of table:



Conclusion: In this Lab report, we learnt about how to connecting mysql in ubuntu and this is very useful for our site, such as programming, creating project and store huge elements of data in database. In all the above tasks we are doing with mysql.

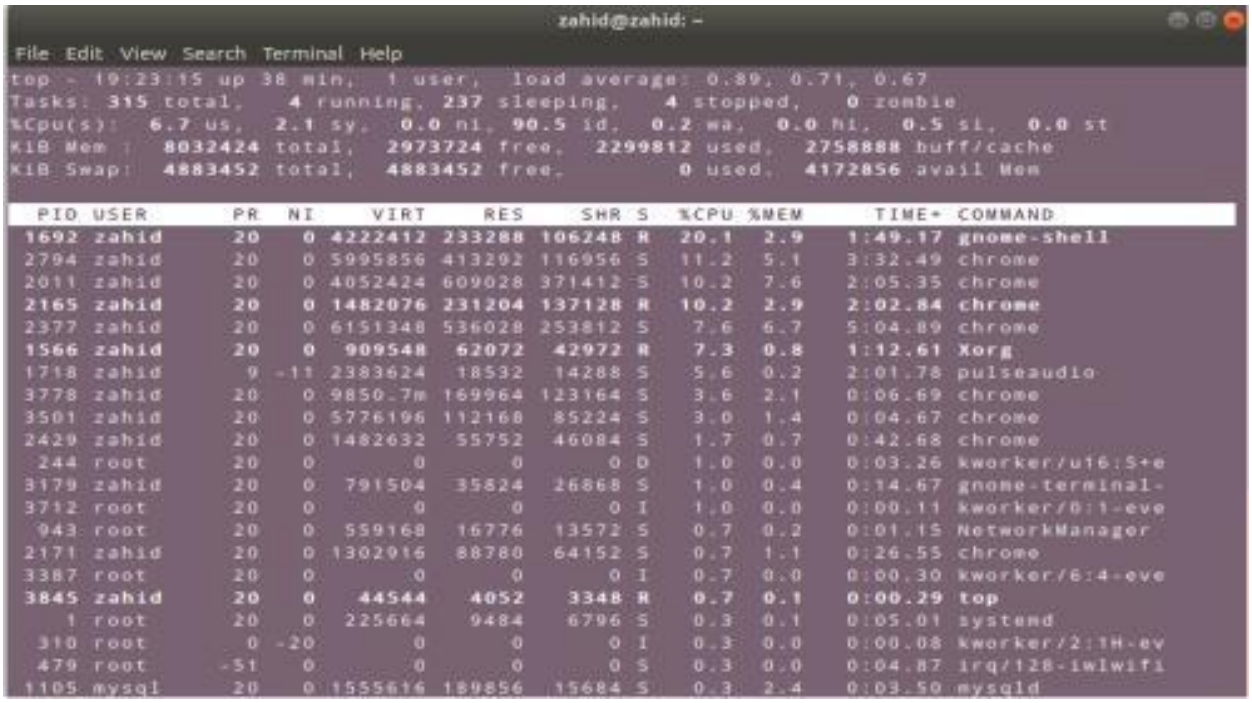
_**Reference:** Zahid Hasan (IT-18017)

Lab No : 06

Name of the Lab Report: Linux command

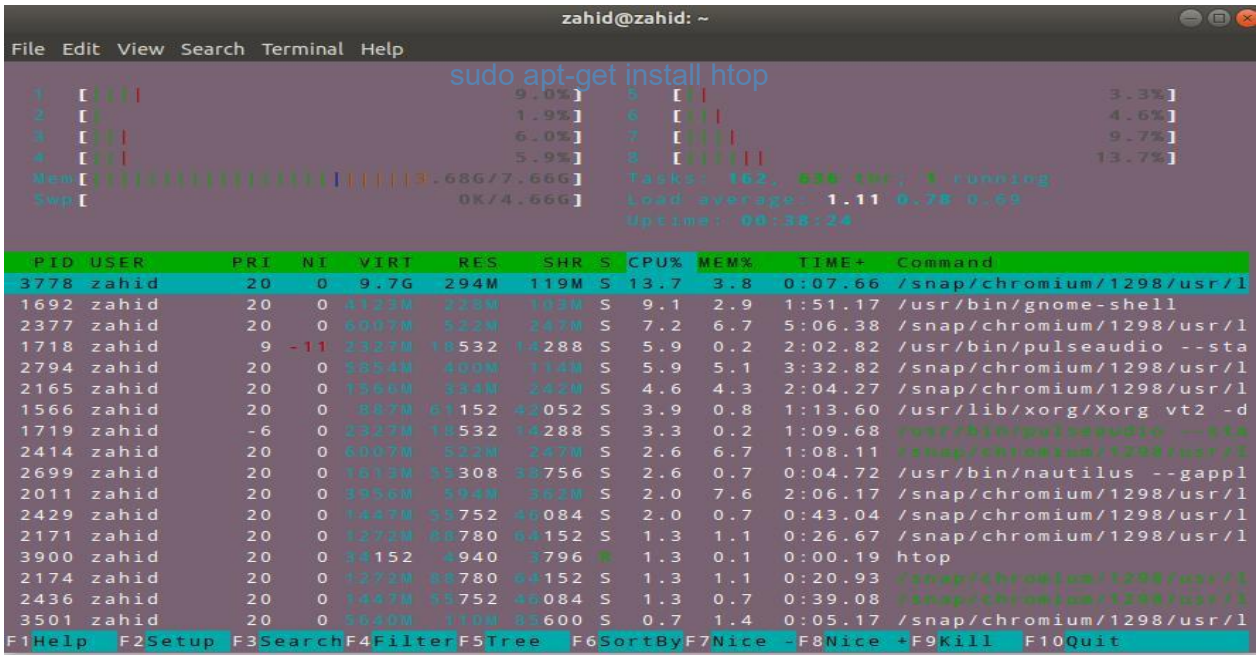
for process

1. top: The top command is the traditional way to view your system's resource usage and see the processes that are taking up the most system resources. Top displays a list of processes, with the ones using the most CPU at the top.



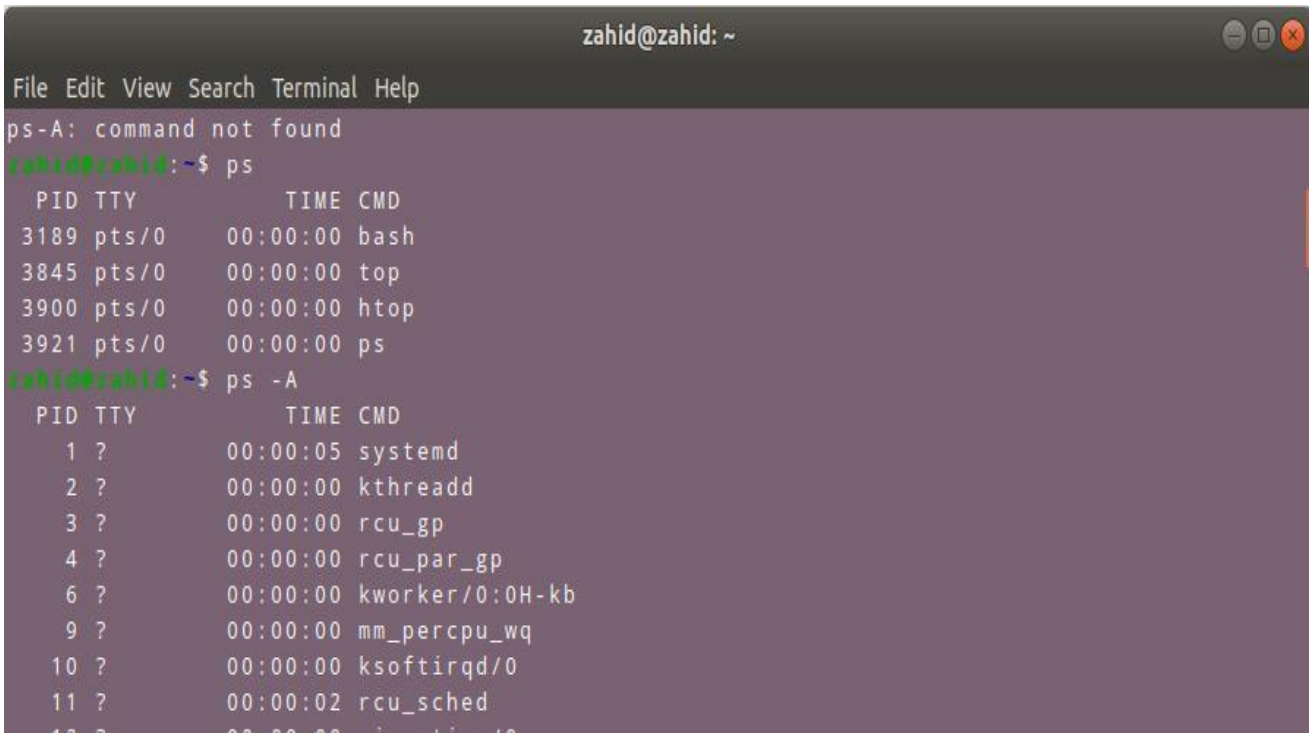
To exit top or htop, use the Ctrl- C keyboard shortcut. This keyboard shortcut usually kills the currently running process in the terminal.

2. htop: The **htop** command is an improved top. It's not installed by default on most Linux distributions — here's the command you'll need to install it on Ubuntu:



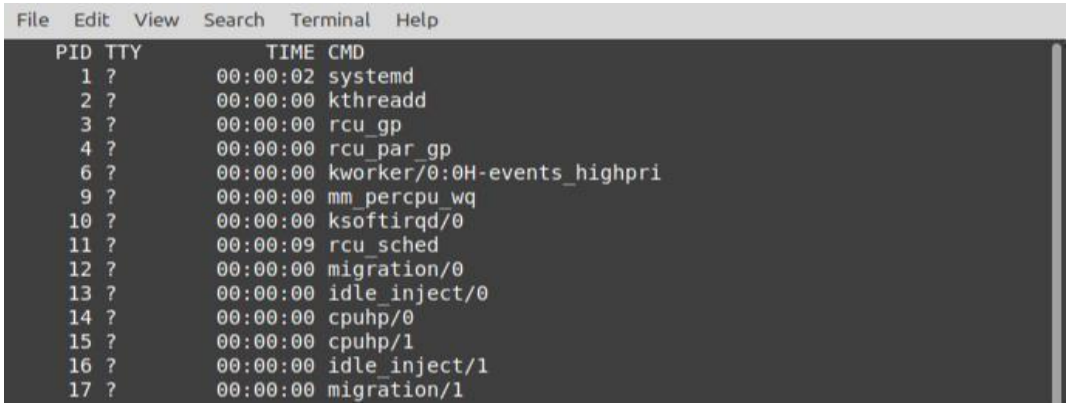
3. ps -A : The **ps** command lists running processes. The following command lists all processes running on your system:

ps -A



4. ps -A | less: ***ps -A*** may be too many processes to read at one time, so we can pipe the output through the **less** command to scroll through them at own pace.

ps -A | less:

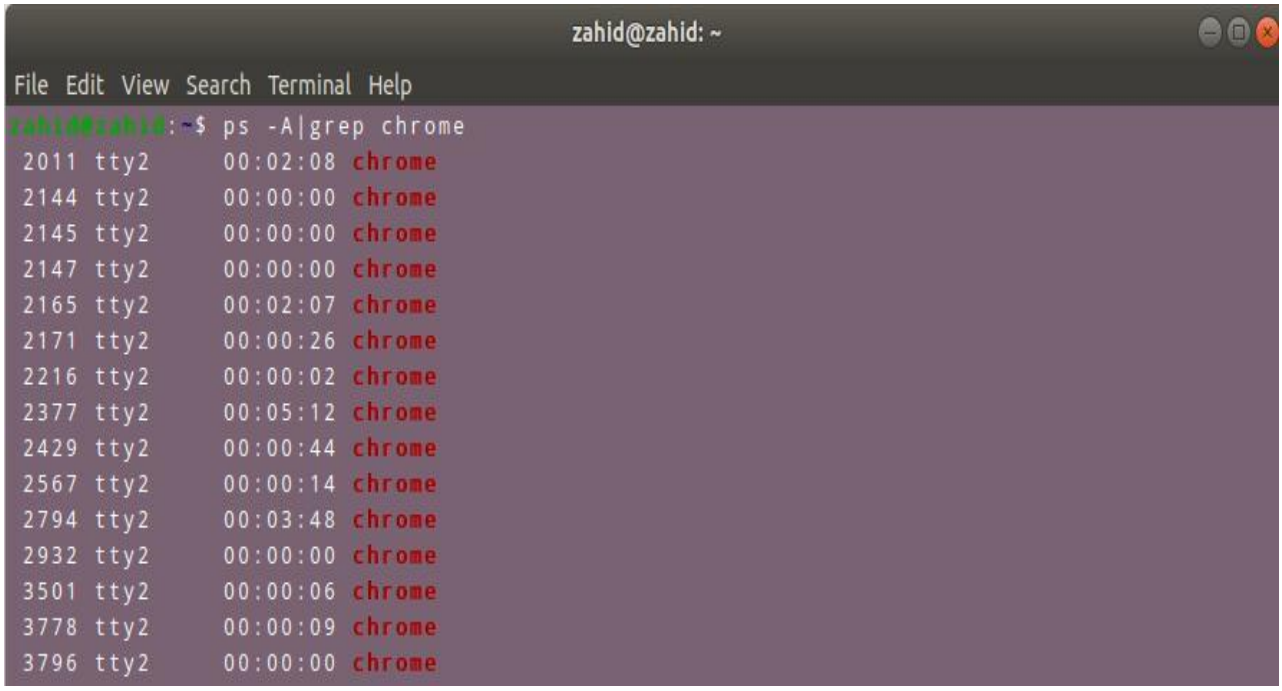


Press q to exit when you're done.

5. ps -A | grep : We could also pipe the output through **grep** to search for a specific process without

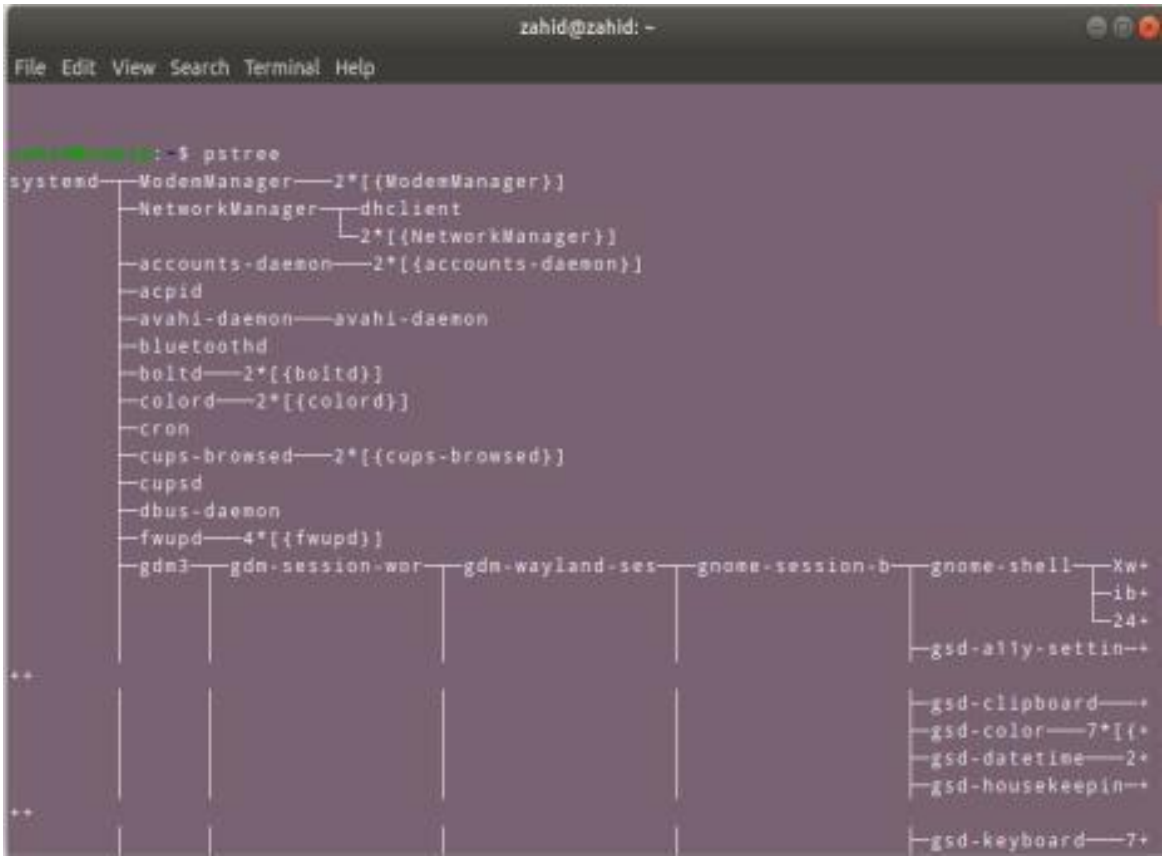
using any other commands. The following command would search for the Firefox process:

ps -A | grep firefox



6. pstree:

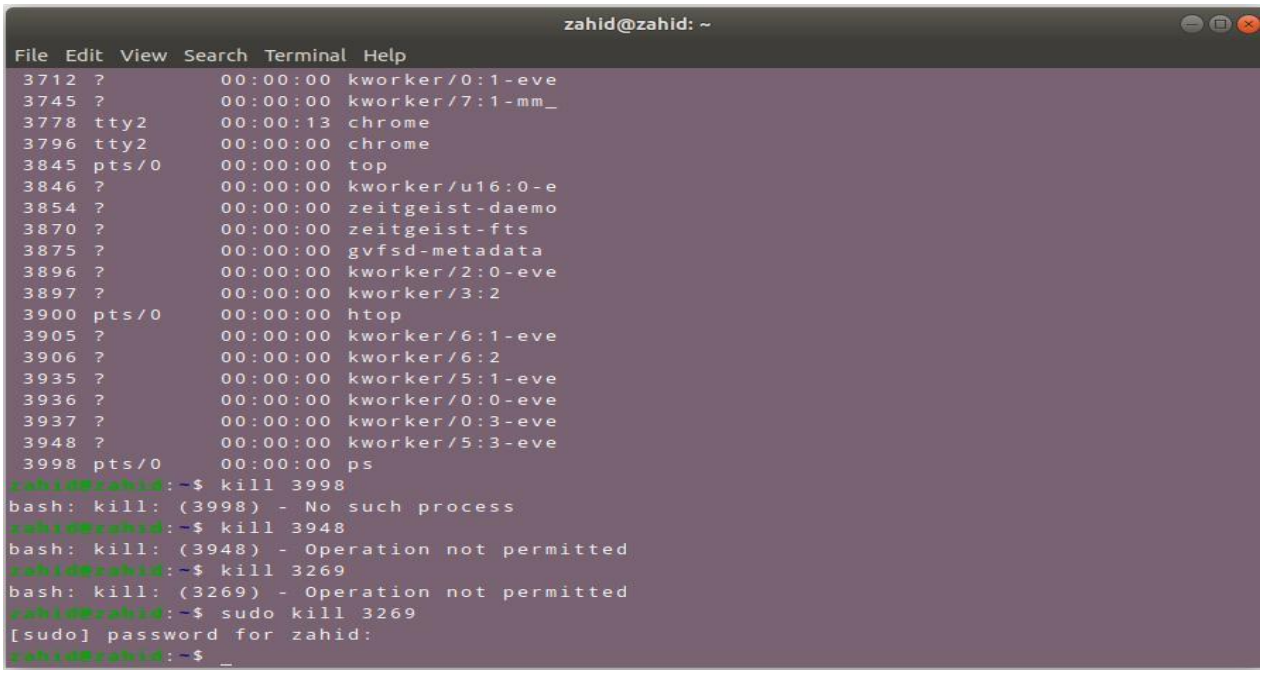
The **pstree** command is anoher way of visualizing processes. It displays them in tree format



7. kill :

The **kill** command can kill a process, given its process ID. You can get this information from the **ps -A**, **top** or **pgrep** commands.

kill PID



8. r enice:

The **renice** command changes the nice value of an already running process. The nice value determines what priority the process runs with. A value of **-19** is very high priority, while a value of **19** is very low priority. A value of **0** is the default priority.

The renice command requires a process's PID. The following command makes a process run with very low priority:

renice 19 *PID*

Conclusion: By doing this lab report, I learnt about the basic concept of thread, thread types, how it can be implemented and also how it works in operating system by using kernel. Mainly, threads are used multiple application running at a same time period in a processor. The main benefit of using thread is that we can do multiple task by dividing a process into multiple threads.

Lab Report No : 07

Name of the Lab Report : Implementation of FCFS Scheduling Algorithm.

Objective: FCFS algorithm Definition and executable code in c are followed int this report.

1. What is FCFS Scheduling algorithm?

Answer: First come, first served (FCFS) is an operating system process scheduling algorithm and a network routing management mechanism that automatically executes queued requests and processes by the order of their arrival. With first come, first served, what comes first is handled first; the next request in line will be executed once the one before it is complete.

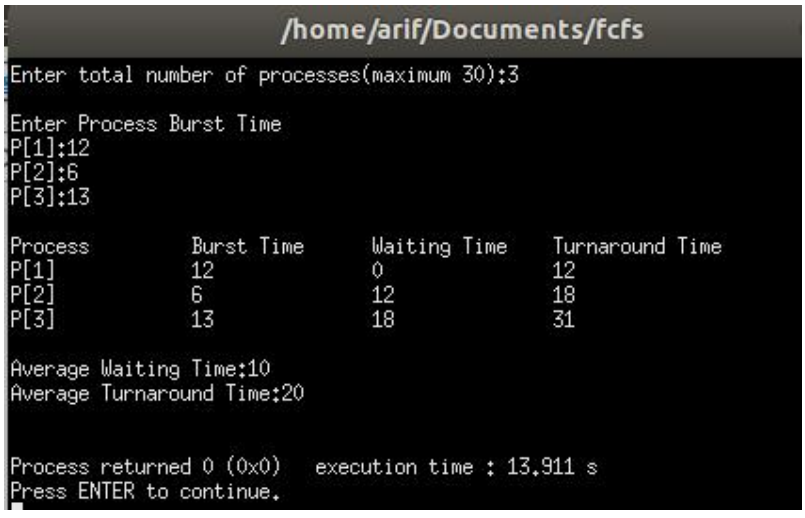
2.How to implemented in C?
Answer:

The code written in c are given below:

```
#include<stdio.h>

int main()

{

        int n,BuT[31],WaT[31],TuT[31],Avwt=0,Avtat=0,i,j;
        printf("Enter total number of processes(maximum
        30):"); scanf("%d",&n);

        printf("\nEnter Process Burst Time\n");

        for(i=0; i<n; i++) {

                printf("P[%d]:",i+1);

                scanf("%d",&BuT[i]);

        }

        WaT[0]=0;

       for(i=1; i<n; i++) {

                WaT[i]=0;

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

                        WaT[i] += BuT[j];
```

```
    }
    printf("\nProcess\t\tBurst Time\tWaiting Time\tTurnaround Time");
    for(i=0; i<n; i++) {
        TuT[i]=BuT[i]+WaT[i];
        Avwt+=WaT[i];
        Avtat+=TuT[i];
        printf("\nP[%d]\t\t%d\t\t%d\t\t%d",i+1,BuT[i],WaT[i],TuT[i]);
    }
    Avwt/=i;
    Avtat/=i;
    printf("\n\nAverage Waiting Time:%d",Avwt);
    printf("\nAverage Turnaround
    Time:%d\n\n",Avtat);
    return 0;
}
```

Output:



Conclusion: In this algorithm, we leart about FCFS algorithm details. Here if we can input a specific number of burst time then we get the output of average waiting time and average Turnaround time.

Lab Report No: 08

Name of the Lab Report: Implementation of SJF Scheduling Algorithm

Objective: SJF algorithm Definition & executable code in C.

1. What is SJF Scheduling Algorithm?

Answer: Shortest job first is a scheduling algorithm in which the process with the smallest execution time is selected for execution next. Shortest job first can be either preemptive or non-preemptive. Owing to its simple nature, shortest job first is considered optimal. It also reduces the average waiting time for other processes awaiting execution.

2. How to implemented in C?

Answer:

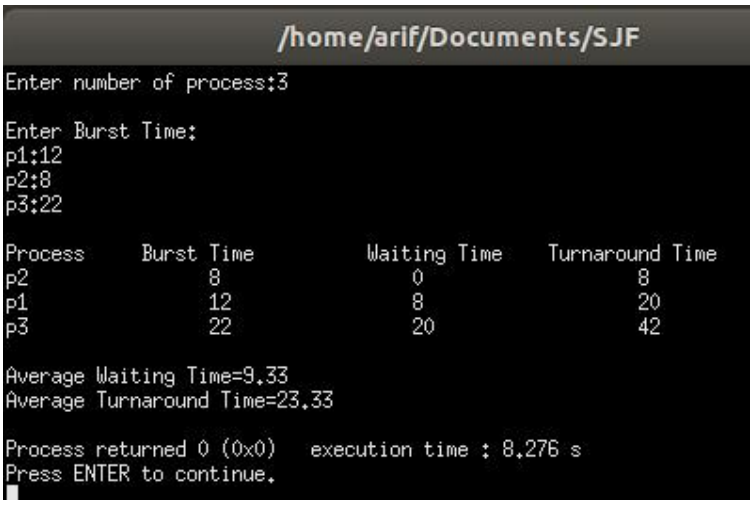
Source Code of SJF Algorithm:

```
#include<stdio.h>

int main()

{

        int BuT[31],Store[31],WaT[31],TaT[31],i,j,n,total=0,pos,temp;

        float Avgwt,AvgTaT;

        printf("Enter number of process:");

        scanf("%d",&n);

        printf("\nEnter Burst Time:\n");

        for(i=0; i<n; i++) {

            printf("p%d:",i+1);

            scanf("%d",&BuT[i]);

            Store[i]=i+1;

        }

        for(i=0; i<n; i++) {

            pos=i;

            for(j=i+1; j<n; j++) {
```

```
            if(BuT[j]<BuT[pos])
                pos=j;
        }
        temp=BuT[i];
        BuT[i]=BuT[pos];
        BuT[pos]=temp;
        temp=Store[i];
        Store[i]=Store[pos];
        Store[pos]=temp;
    }
    WaT[0]=0;
    for(i=1; i<n; i++) {
        WaT[i]=0;
        for(j=0; j<i; j++)
      WaT[i]+=BuT[j];
        total+=WaT[i];
    }
    Avgwt=(float)total/n;
    total=0;
    printf("\nProcess\t    Burst Time       \tWaiting Time\tTurnaround Time");

    for(i=0; i<n; i++) {
        TaT[i]=BuT[i]+WaT[i];
        total+=TaT[i];
        printf("\np%d\t\t             %d\t\t
                            %d\t\t\t%d",Store[i],BuT[i],WaT[i],TaT[i]);
    }
    AvgTaT=(float)total/n;
    printf("\n\nAverage Waiting Time=%.2f",Avgwt);
    printf("\nAverage Turnaround Time=%.2f\n",AvgTaT);
    return 0;
```

```
}
```

Output:


/home/arif/Documents/SJF
Enter number of process:3
Enter Burst Time:
p1:12
p2:8
p3:22
Process Burst Time Waiting Time Turnaround Time
p2 8 0 8
p1 12 8 20
p3 22 20 42
Average Waiting Time=9.33
Average Turnaround Time=23.33
Process returned 0 (0x0) execution time : 8.276 s
Press ENTER to continue.

Conclusion: In this lab, we learnt about SJF algorithm. We are implementing this algorithm with C programming language. The main advantage of this algorithm is if we give the processes and burst time value, then it returns the average waiting time and average turnaround time.

Lab Report No: 09

Name of the Lab Report: Implementation of Priority Scheduling Algorithm

Objective: Priority Scheduling algorithm Definition & executable code in c are followed.

1. What is priority Scheduling algorithm?

Answer: Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems. Each process is assigned a priority. Process with highest priority is to be executed first and so on. Processes with same priority are executed on first come first served basis. Priority can be decided based on memory requirements, time requirements or any other resource requirement.

2. How to implemented in C?

Answer:

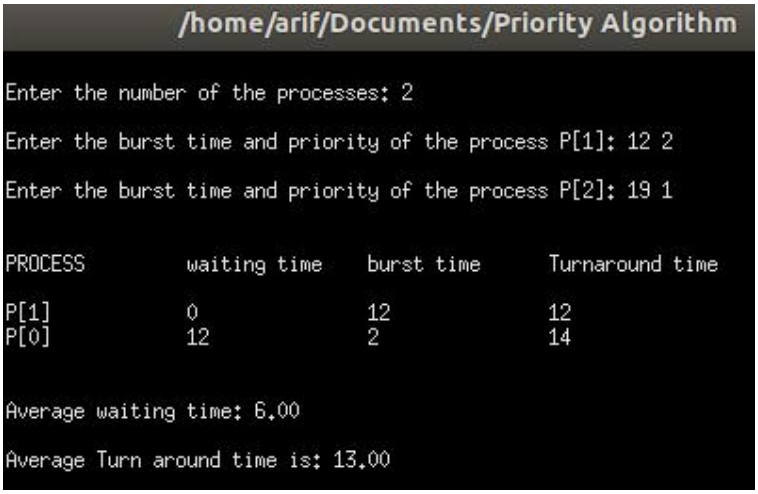
Source Code:

```
#include <stdio.h>

int main()

{

    int BuT[20],WaT[20],p[20],TaT[20],priority[20];

    float avwt=0,avtat=0;

    int i,j,n,temp,key;

    printf("\nEnter the number of the processes: ");

    scanf("%d",&n);

    for(i=1;i<=n;i++)



    {
```

```
    printf("\nEnter the burst time and priority of the process P[%d]: ",i);
    scanf("%d",&BuT[i]);
    scanf("%d",&priority[i]);
    p[i]=i;
}
for(i=0;i<n;i++)
{
    key=i;
    for(j=i+1;j<n;j++)
    {
        if(priority[j]<priority[key])
        {
            key=j;
        }
    }
    temp=BuT[i];
    BuT[i]=BuT[key];
    BuT[key]=temp;
    temp=priority[i];
    priority[i]=priority[key];
    priority[key]=temp;
    temp=p[i];
    p[i]=p[key];
```

```
        p[key]=temp;
    }
    WaT[0]=0;
    TaT[0]=BuT[0];
    avtat=TaT[0];
    for(i=1;i<n;i++)
    {
            WaT[i]=WaT[i-1]+BuT[i-1];
            TaT[i]=TaT[i-1]+BuT[i];
            avwt+=WaT[i];
            avtat+=TaT[i];
    }
    avwt=avwt/n;
    avtat=avtat/n;
    printf("\n\nPROCESS\t\twaiting time\tburst time\tTurnaround time\n");
    printf("\n");

    for(i=0;i<n;i++)
{
       printf("P[%d]\t\t%d\t\t%d\t\t%d\n",p[i],WaT[i],BuT[i],TaT[i]);
       }
       printf("\n\nAverage waiting time: %.2f",avwt);
       printf("\n\nAverage Turn around time is:
       %.2f",avtat); printf("\n");

       return 0;
}
```

Output:



Conclusion: In this lab, we learnt about Priority Scheduling Algorithm. The algorithm is implemented with C programming language. In this algorithm, if we continously give processes and burst time, it returns the average waiting time and turnaround time. And this tasks are useful in linux.

Lab Report No: 10

Name of the Lab Report: Implementation of Round Robin Scheduling Algorithm.

Objective: Round Robin Scheduling algorithm Definition & executable code in c are followed.

1. What is Round Robin Scheduling algorithm?

Answer: Round robin scheduling is the preemptive scheduling in which every process get executed in a cyclic way, i.e. in this a particular time slice is allotted to each process which is known as time quantum. Every process, which is present in the queue for processing, CPU is assigned to that process for that time quantum. Now, if the execution of the process gets completed in that time quantum, then the process will get terminate otherwise the process will again go to the ready queue, and the previous process will wait for the turn to complete its execution.

The scheduling drives its name from the principle which is known as a round robin in which every person takes an equal share of anything they have in turn. We make use of round robin scheduling algorithm in a time-sharing system. It is generally used by those os which has multiple clients to make use of resources.

2. How to implemented in C?
Answer:

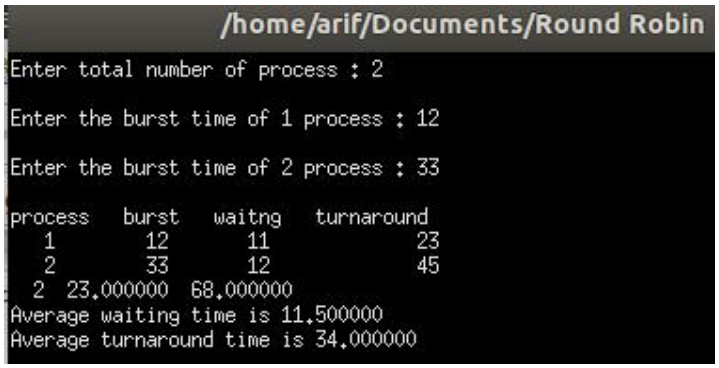
Source Code:

```
#include<stdio.h>

int main()

{

 int bursttime[100],waitingtime[100],turnaroundtime[100],b[100];

 int i,n,time,count=0;

 float totalwt=0,totalTT=0,avgwt,avgtt;



 printf("Enter total number of process : ");

 scanf("%d",&n);

 for(i=0;i<n;i++)

 {

  printf("\nEnter the burst time of %d process : ",i+1);

  scanf("%d",&bursttime[i]);
```

```
 b[i] = bursttime[i];
}
i=0;
for(time=0;count!=n;time++)
{
  while(bursttime[i] == 0)
  {
    i=(i+1)%n;
  }
  bursttime[i]--;
  if(bursttime[i]==0)
  {
    turnaroundtime[i] = time+1;
    count++;
  }
  i = (i+1)%n;
}
printf("\nprocess  burst  waitng  turnaround  ");
for(i=0;i<n;i++)
{  waitingtime[i] = turnaroundtime[i] - b[i];
   printf("\n  %d \t   %d \t    %d \t   %d",i+1,b[i],waitingtime[i],turnaroundtime[i]);
   totalwt = totalwt + waitingtime[i];
   totalTT = totalTT + turnaroundtime[i];
}
printf("\n  %d  %f  %f",n,totalwt,totalTT);
avgwt = totalwt / n;
avgtt = totalTT / n;
printf("\nAverage waiting time is %f",avgwt);
```

```
 printf("\nAverage turnaround time is %f ",avgtt);

 return 0;

}
```

Output:



Conclusion: In this lab, we have implemented round robin scheduling algorithm with C language. This algorithm is mainly depends on preemptive and quantum in nature. Its very helpful to understand and realize round robin algorithm.

Lab Report No: 11

Name of the Lab Report: Implementation of FIFO page replacement Algorithm

Objective: FIFO page replacement algorithm Definition & executable code in c are followed.

1. What is FIFO page replacement algorithm?

Answer: This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal

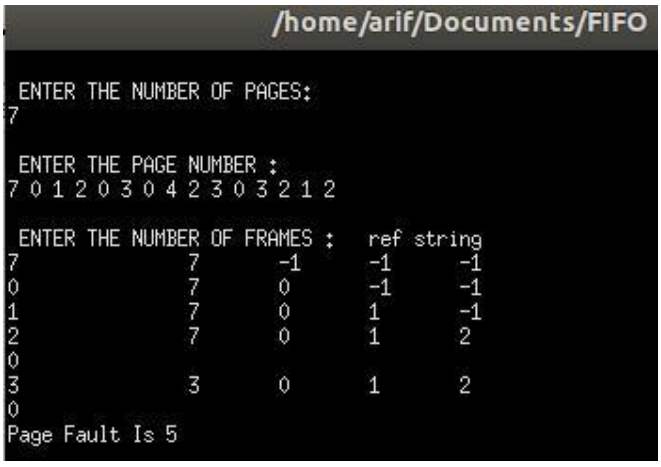
2. How to implemented in C? Answer:

Source Code:

```
#include<stdio.h>

int main()

{

int i,j,n,a[50],frame[10],no,k,avail,count=0;

printf("\n ENTER THE NUMBER OF PAGES:\n");

scanf("%d",&n);

printf("\n ENTER THE PAGE NUMBER :\n");

for(i=1;i<=n;i++)

scanf("%d",&a[i]);

printf("\n ENTER THE NUMBER OF FRAMES :");

scanf("%d",&no);

for(i=0;i<no;i++)

      frame[i]= -1;

      j=0;

      printf("\tref string\t page frames\n");

for(i=1;i<=n;i++)
```

```
{
    printf("%d\t\t",a[i]);
    avail=0;
    for(k=0;k<no;k++)
if(frame[k]==a[i])
  avail=1;
   if (avail==0)
    {
          frame[j]=a[i];
         j=(j+1)%no;
          count++;
        for(k=0;k<no;k++)
            printf("%d\t",frame[k]);
}
       printf("\n");
}
  printf("Page Fault Is %d",count);
      return 0;
}
```

Output:



Conclusion: In this Lab report we learnt about FIFO page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, yhe oldest page in the front of the queue. When a page needs to be replaced page in front of the queue is selected for removal. For our better understand we have used C language.