

# LINUX

My\_WoRk\_L<sup>A</sup>T<sub>E</sub>X

HEMANT GORAKSH GHUGE



April 22, 2018

first created on 8 June 2017

last modified on 22 June 2017

# Contents

<b>1</b>	<b>Ubuntu-Desktop</b>	<b>1</b>
1.1	Learning Objectives . . . . .	1
1.2	Terminal(command line) . . . . .	1
1.3	Firefox Web Browser . . . . .	1
1.4	Desktop . . . . .	1
1.5	Summary . . . . .	1
<b>2</b>	<b>Desktop-Customization</b>	<b>2</b>
2.1	Learning Objectives & Summary . . . . .	2
<b>3</b>	<b>Synaptic-Package-Manager</b>	<b>3</b>
<b>4</b>	<b>Ubuntu-Software-Center</b>	<b>4</b>
<b>5</b>	<b>Basic-Commands</b>	<b>5</b>
5.1	What are Commands? . . . . .	5
5.2	Command Interpreter / Shell . . . . .	5
5.3	Types of Commands . . . . .	5
5.3.1	External Commands . . . . .	5
5.3.2	Internal Commands . . . . .	6
5.4	Structure of Commands . . . . .	6
5.5	man Command . . . . .	6
<b>6</b>	<b>General Purpose Utilities</b>	<b>7</b>
6.1	Common escape sequences with echo command . . . . .	7
6.2	The Root User . . . . .	7
<b>7</b>	<b>File System</b>	<b>9</b>
7.1	File . . . . .	9
7.2	Directories . . . . .	9
7.3	File Inode . . . . .	9
7.4	Types of Files . . . . .	9
7.5	All files in Linux are related . . . . .	10
7.6	Home directory and Current directory . . . . .	10
7.7	Change Directory(cd) . . . . .	10
7.8	The rmdir command . . . . .	11
7.9	The rmdir command . . . . .	11
<b>8</b>	<b>Working with Regular Files</b>	<b>12</b>
8.1	The cp command . . . . .	12
8.2	The cp command option . . . . .	12
8.3	The mv command . . . . .	12
8.4	The mv command options . . . . .	13

8.5	The rm command . . . . .	13
8.6	The cmp command . . . . .	13
8.7	The wc command . . . . .	13
<b>9</b>	<b>File Attributes</b>	<b>14</b>
9.1	What is File Attribute? . . . . .	14
9.2	Changing Ownership . . . . .	14
9.3	chmod command . . . . .	14
9.4	File Permissions . . . . .	15
9.5	Changing Group . . . . .	15
9.6	Inode in Linux . . . . .	15
9.7	Hard Links . . . . .	16
9.8	Soft Links . . . . .	16
<b>10</b>	<b>Redirection &amp; Pipes</b>	<b>17</b>
10.1	Stream : . . . . .	17
10.2	File Descriptor : . . . . .	17
10.3	The three standard streams . . . . .	17
10.3.1	stdin . . . . .	17
10.3.2	stdout . . . . .	17
10.3.3	stderr . . . . .	17
10.4	More about Streams . . . . .	18
10.5	Redirection- Two ways . . . . .	18
10.5.1	n> . . . . .	18
10.5.2	n>> . . . . .	18
10.6	Pipes . . . . .	19
<b>11</b>	<b>Working with Linux Process</b>	<b>20</b>
11.1	What is a process? . . . . .	20
11.2	Shell process . . . . .	20
11.3	Spawning . . . . .	20
11.4	Process attributes . . . . .	21
11.5	ps command . . . . .	21
11.6	Process types . . . . .	21
11.6.1	User processes . . . . .	21
11.6.2	System processes . . . . .	21
11.7	All processes . . . . .	21
<b>12</b>	<b>The Linux Environment</b>	<b>22</b>
12.1	Shell Variables . . . . .	22
12.1.1	Environment Variables . . . . .	22
12.1.2	Local Variables . . . . .	22
<b>13</b>	<b>Basics of System Administration</b>	<b>24</b>
13.1	Learning Objectives . . . . .	24
13.2	Prerequisite . . . . .	24
13.3	adduser . . . . .	24
13.4	sudo . . . . .	24
13.5	su . . . . .	25
13.6	usermod . . . . .	25
13.7	id . . . . .	25
13.8	userdel . . . . .	25
13.9	df and du . . . . .	25

13.10 Summary . . . . .	26
<b>14 Simple Filters</b>	<b>27</b>
14.1 Prerequisite . . . . .	27
14.2 head command . . . . .	27
14.3 tail command . . . . .	27
14.4 Common Linux log files names & Usage . . . . .	27
14.5 sort command . . . . .	28
14.6 cut command . . . . .	28
14.7 paste command . . . . .	29
<b>15 The grep command</b>	<b>30</b>
15.1 Regular Expression . . . . .	30
15.2 Introduction to grep . . . . .	30
15.3 Summary . . . . .	31
15.4 Assignment . . . . .	31
<b>16 More on grep command</b>	<b>32</b>
16.1 Regular Expressions . . . . .	32
16.2 The Character Class [] . . . . .	32
16.3 Range . . . . .	32
16.4 The Asterisk . . . . .	32
16.5 Summary . . . . .	33
16.6 Assignment . . . . .	33
<b>17 The sed command : The stream editor</b>	<b>34</b>
17.1 Introduction to sed . . . . .	34
17.2 Line & Context addressing . . . . .	35
17.3 Summary . . . . .	35
17.4 Assignment . . . . .	35
<b>18 More on sed command</b>	<b>36</b>
18.1 Substitution & Replacement . . . . .	36
18.2 Insertion . . . . .	36
18.3 Summary . . . . .	36
18.4 Assignment . . . . .	36
<b>19 Basics of awk</b>	<b>37</b>
19.1 Introduction . . . . .	37
19.2 Parameters . . . . .	38
19.3 Summary . . . . .	38
19.4 Assignment . . . . .	38

# Chapter 1

## Ubuntu-Desktop

### 1.1 Learning Objectives

1. ULD on gnome
2. Some applications in Ubuntu Desktop
3. Changing the theme of the Desktop

### 1.2 Terminal(command line)

In fact, it is more powerful than the GUI

\$ ls	list
-------	------

### 1.3 Firefox Web Browser

F6 - go to address bar  
<http://spoken-tutorial.org/>

### 1.4 Desktop

*ctrl + windows + D*

### 1.5 Summary

1. Ubuntu Desktop
2. The launcher and some icon visible on it
  - (a) Application
  - (b) Calculator
  - (c) Text Editor
  - (d) Movie Player
  - (e) LibreOffice Suite

# Chapter 2

## Desktop-Customization

### 2.1 Learning Objectives & Summary

1. about the launcher
2. remove and add application in the launcher
3. use multiple desktop
4. internet connectivity
5. sound settings
6. date & time settings
7. switch to other user account

# Chapter 3

## Synaptic-Package-Manager

1. It is not a pre-installed application, has to be downloaded from Ubuntu Software Center.
2. Install, remove and upgrade software packages in user friendly way.
3. Synaptic is graphical package management tool based on GTK+ and APT.
4. Besides this basic function these features are provided:
  - (a) Search and filter the list of available packages
  - (b) Perform smart system upgrades
  - (c) Fix broken package dependencies
  - (d) Edit the list of used repositories(source.list)
  - (e) Download the latest changelog of a package
  - (f) Configure packages through the debconf system
  - (g) Browse all available documentation related to a package(dwwww is required)

# Chapter 4

## Ubuntu-Software-Center

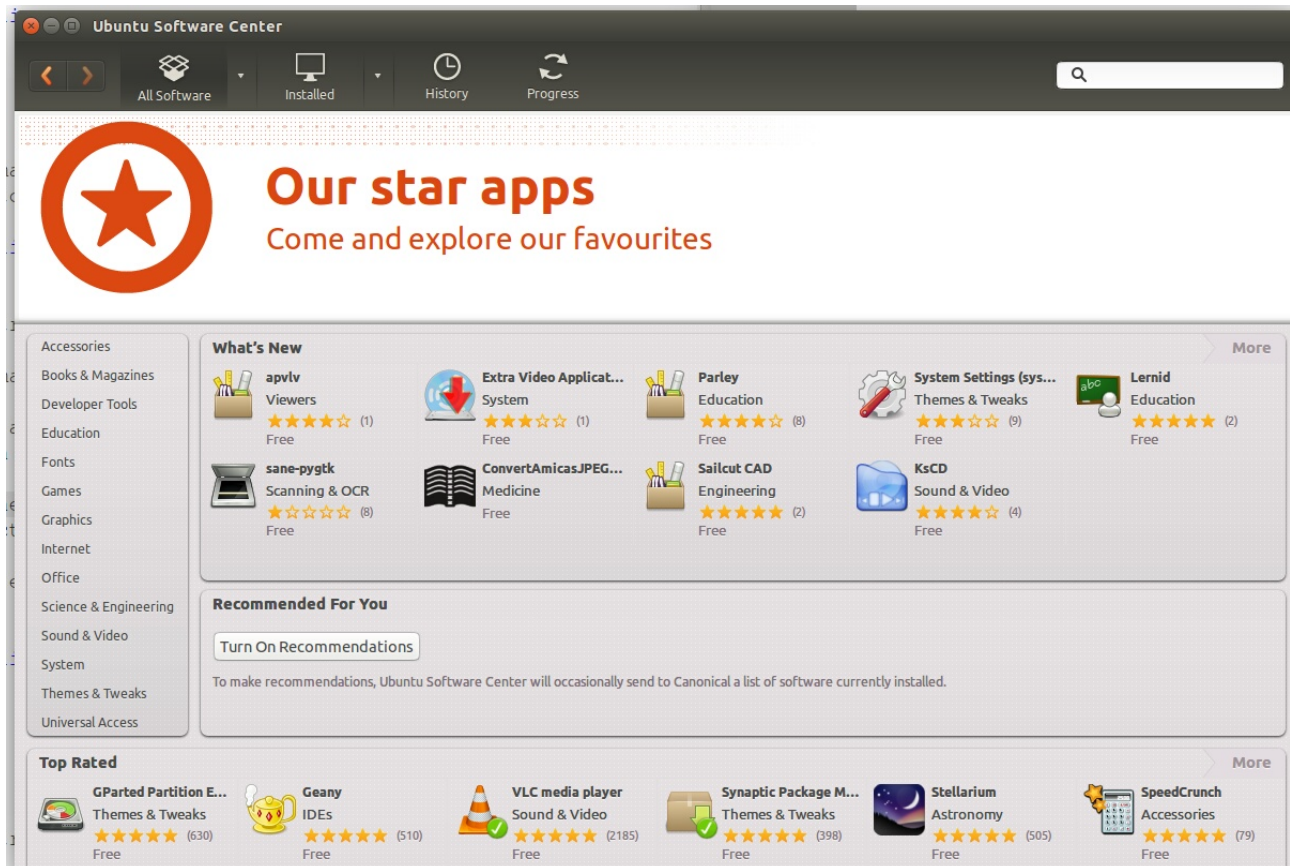


Figure 1: Ubuntu-Software-Center



# Chapter 5

## Basic-Commands

### 5.1 What are Commands?

1. words that when keyed in cause some action take place
2. Seldom more than four characters in length like ls, who, ps
3. Lower case
4. Case sensitive

\$ who      who is logged in
------------------------------

### 5.2 Command Interpreter / Shell

1. Program that act as the interface between us and the Linux system
2. Allow us to enter command for the operating system to execute
3. Multiple shell
4. Standard shell in Linux
  - (a) Bash shell
5. Other shells
  - (a) Bourne shell (sh)
  - (b) C shell (csh)
  - (c) Korn shell (ksh)

\$ echo \$SHELL      to see which shell is used
\$ type ps            this shows where ps command is stored
\$ echo \$PATH

### 5.3 Types of Commands

#### 5.3.1 External Commands

1. separate files/program
2. Eg : most commands like ls, ps etc

### 5.3.2 Internal Commands

1. Implementation written within the shell
2. Do not exist as separate file
3. Eg : echo etc

```
$ type echo      echo is shell builtin
```

## 5.4 Structure of Commands

1. One word or multiple words separated from white spaces
2. Multiple word case
  - (a) First word - the actual name of the command
  - (b) Other words - arguments
3. Arguments - options or expression or filenames
4. A command can perform different task depending on the option specified
5. Short and long option

```
$ ls
$ ls -a
$ ls -d
$ ls -all
```

## 5.5 man Command

1. System's Manual Pager
2. Argument - name of a program, utility or function.

```
$ man man
$ man -k directories    if confused which command to use
$ apropos directories    copy above
$ whatis ls
$ ls --help
```

# Chapter 6

## General Purpose Utilities

```
$ echo Hello World
$ echo $SHELL
```

### 6.1 Common escape sequences with echo command

1. We need to use the -e option of echo command for using escape sequences
  - (a) \t for tab
  - (b) \n for newline
  - (c) \c for displaying prompt on the same line after message has been echoed

```
$ uname -r      what Kernel version you are using
$ who am I      user name
$ passwd        password change
```

### 6.2 The Root User

1. He is a special user with extra privileges
2. A root user is similar to a user in Windows with Administrator status

```
$ date
$ date +%T
$ date +%h
$ date +%m
$ date +%y
$ date +%''%h%y''

$ cal
$ cal 12 2017
```

3. command 'pwd' & 'echo \$HOME' are used for same thing i.e. present working directories

```
$ ls
$ ls -all      all files including hidden
$ ls -l        file permission, file owner name, last modification time, file size etc.
$ ls -l > fileinfo  redirects info into filename fileinfo
```

```
$ cat info
```

```
$ cat > file
```

then write what you want to save

for save&quit ctrl+d

for only quit ctrl+c

```
$ cat >>file
```

# Chapter 7

## File System

### 7.1 File

1. In real life we know that a file is where we store our documents and papers.
2. Similarly in Linux a file is a container for storing information.

### 7.2 Directories

1. A collection of files and other (sub)directories.
2. It gives:
  - (a) Systematic storage of files.
  - (b) Better protection and access control
  - (c) Easier naming scheme for files

### 7.3 File Inode

1. Along with its contents, a file has a name and some properties, like:
  - (a) The file's creation/modification date, owner, size, its permissions where on the disk it's stored
2. The properties are stored in the file's inode, a special block of data in the system.
3. A directory is a file that holds the inode numbers and names of other files.

### 7.4 Types of Files

1. In Linux there are three kinds of files:
  - (a) Regular Files or Ordinary files
  - (b) Directories
  - (c) Device Files
    - i. helps to read from and write to devices in a way similar to that for ordinary files.

## 7.5 All files in Linux are related

1. A directory containing files and subdirectories have a parent child relationship with each other.
2. Linux File System Tree
3. At the top is the root (denoted by frontslash /).
4. Easy navigation from one file or directory to other.

## 7.6 Home directory and Current directory

1. When we login into the Linux system we are by default in a home directory.
2. The pwd command helps us to see the current directory.

```
$ echo $HOME  
$ pwd
```

## 7.7 Change Directory(cd)

1. We can move from one directory to other. The cd command is used for this.

```
$ cd /usr  
$ pwd
```

2. Absolute Pathnames and Relative Pathnames
3. . represents current directory
4. .. represents the parent of the current directory

```
$ cd  
$ cd Music  
$ pwd  
$ cd ..  
$ pwd  
$ cd ../Documents/  
$ pwd  
$ cd  
$ pwd
```

5. root

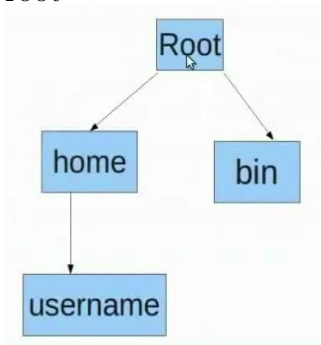


Figure 2: root

6. How to change directory to bin?

```
$ cd ../../bin
$ pwd
$ cd
```

## 7.8 The mkdir command

Used to create a directory

```
$ mkdir testdir
$ mkdir test1 test2
$ mkdir testtree testtree/test3
```

## 7.9 The rmdir command

1. Used to removing a directory or directories.
2. A directories can be removed by us only
  - (a) if we are its owner
  - (b) our current directory is hierarchically above the directory to be removed
  - (c) the directory is empty

```
$ rmdir test1
$ cd testtree/test3
$ rmdir testdir
$ cd ..
$ cd ..
$ rmdir testdir
$ rmdir testtree testtree/test3
$ cd testtree
$ ls
```

# Chapter 8

## Working with Regular Files

### 8.1 The cp command

1. To copy a file from one place to another

```
$ cp [OPTION]... SOURCE DEST
$ cp [OPTION]... SOURCE(s)... DIRECTORY
```

2. It copies SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY

```
$ cat test1
$ cp test1 test2
$ cat test2
$ cp /home/anirban/arc/demo1 /home/anirban/arc/demo2
$ ls /home/anirban
$ cp /home/anirban/arc/demo1 /home/anirban/
$ ls /home/anirban
$ cp test1 test2 test3 /home/anirban/testdir
$ ls /home/anirban/testdir
```

### 8.2 The cp command option

1. -R: For recursively copying directories

```
$ cp testdir/ test
error
$ cp -R testdir/ test
$ ls
$ clear
$ ls test
```

2. -b: For backup
3. -i: (interactive), warns us before overwriting any destination file

### 8.3 The mv command

1. This is used for moving files.
2. It has two major uses.
  - (a) Rename a file or directory



- (b) Moves a group of files to different directory.

```
$ mv test1 test2
$ mv -i anirban test2
$ ls
$ mv abc.txt pop.txt push.txt testdir
$ ls testdir
```

## 8.4 The mv command options

1. -b: For backup, it will backup every file in the destination before it is overwritten.
2. -i: (interactive), warns us before overwriting any destination file

## 8.5 The rm command

1. This command is used for deleting files.
2. It can delete a single file as in

```
$ rm testdir/faq.txt
```

3. or multiple files as in:

```
$ rm testdir/abc1 testdir/abc2
```

4. Options

- (a) -r may be right protected
- (b) -f force delete

```
$ rm testdir
error rm: cannot remove
$ rm -rf testdir
```

## 8.6 The cmp command

1. Compares two files byte by byte.
  - (a) Same content -> no message
  - (b) Different content -> location of the first mismatch is printed

```
$ cmp file1 file2
$ cat sample1
$ cat sample2
$ cmp sample1 sample2
$ clear
```

## 8.7 The wc command

```
$ cat sample3
$ wc sample3      lines, words, characters
6 67 385
```

# Chapter 9

## File Attributes

### 9.1 What is File Attribute?

1. A file attribute is metadata that describes or is associated with computer file
2. File attribute is the characteristics that describe a file, such as owner, file type, access permissions, etc.

### 9.2 Changing Ownership

1. chown command is used to change the ownership of the file or directory. Only administrator or root user can change the owner of a file or directory.
2. The syntax of chown command is

```
$ chown [option] ownername filename/directoryname
```

3. -R : To change the permission on files that are in the subdirectories of the directory
4. -c : Change the permission for each file.
5. -f : Prevents chown from displaying error messages

```
$ cd Desktop/file_attr  
$ ls -l testchown  
$ sudo su chown -R anusha test_chown  
$ ls -l
```

### 9.3 chmod command

1. chmod command is used to change the access mode (permissions) of one or more files.
2. syntax of the chmod command is

```
$ chmod [options] mode filename
```

3. we may give the following options with chmod command.
4. -c, --changes : Print information about files that are changed.
5. -f, --silent, --quiet : Do not notify user of files that chmod cannot change.

## 9.4 File Permissions

r : Read  
w : Write  
x : Execute  
s : Set user (or group) ID

Alternatively, we may specify permissions by a three-digit octal number.

1st digit : owner permissions  
2nd digit : group permissions  
3rd digit : other permissions

4 : Read  
2 : Write  
1 : Execute

```
$ chmod u+x example1
$ ls -l example1
$ chmod 751 example1
$ ls -l example1
$ chmod =r example1
$ ls -l example1
$ chmod -R 755 directory1
$ ls -l
$ chmod u+x example2
$ ls -l example2
$ chmod g+w example3
$ ls -l example3
$ chmod a-w example3
$ ls -l example3
```

## 9.5 Changing Group

1. chgrp command is used to change the group of one or more files to new group.
2. The syntax for the chgrp command is

```
$ chgrp [options] newgroup files
```

```
$ ls -l example4
$ sudo chgrp rohit example4
$ ls -l example4
```

## 9.6 Inode in Linux

1. The inode number is a unique integer assigned to the device.
2. We can use ls -i command to see the inode number of a file.

```
$ ls -i example5
```

## 9.7 Hard Links

1. Inodes are associated with precisely one directory entry at a time.
2. Hard links are to associate multiple directory entries with a single inode
3. `ln` is the command to make link
4. The syntax of `ln` command to create the hard link is

```
$ ln source link
```

5. Where, source is an existing file and link is the file to create.

```
$ clear  
$ ln example1 exampleln  
$ ls -i example1 exampleln
```

## 9.8 Soft Links

1. Soft link (symbolic link) is a special type of file that contains a reference to another file or directory in the form of an absolute or relative path.
2. The syntax of `ln` command to create the soft link is

```
$ ln -s target-filename symbolic-filename
```

```
$ ln -s example1 examplesoft  
$ ls -li example1 examplesoft
```

# Chapter 10

## Redirection & Pipes

### 10.1 Stream :

1. A Linux shell, such as Bash, receives input and output as sequences or streams of characters.
2. Accessed using file IO techniques.
3. Actual stream of characters may come from or go to a file

### 10.2 File Descriptor :

1. Integer number
2. Associated with every open file of a process

### 10.3 The three standard streams

#### 10.3.1 stdin

1. It is the standard input stream
2. Provides input to commands
3. It has file descriptor 0

#### 10.3.2 stdout

1. It is the standard output stream
2. Provides output to commands
3. It has file descriptor 1

#### 10.3.3 stderr

1. It is the standard error stream
2. Provides error to commands
3. It has file descriptor 2

## 10.4 More about Streams

1. Input streams provide input to programs
2. By default it takes from terminal keystrokes
3. Output streams print text characters
4. By default to the terminal -text window on a graphical desktop
5. we can change this default behaviour - Redirection

\$ wc	
\$ wc < test1.txt	redirect stdin
\$ wc test1.txt	open the file

## 10.5 Redirection- Two ways

### 10.5.1 n>

1. Redirects output from file descriptor n to a file
2. Need write authority to the file
3. Destination file if does not exist - created
4. Destination file if existed - the existing contents are lost without any warning

### 10.5.2 n>>

1. Redirects output from file descriptor n to a file
2. Need write authority to the file
3. Destination file if does not exist - created
4. Destination file if existed - the output is appended to the existing file

IMP.

1. Redirects stdout > or >> (same as) 1> or 1>>
2. Redirects stderr 2> or 2>>

```
# wc test1.txt > wc_results.txt
# cat wc_results.txt
# wc test2.txt > wc_results.txt
# cat wc_results.txt

# wc test1.txt >> wc_results.txt
# cat wc_results.txt

# wc aaa
# wc aaa 2> errorlog.txt
# cat errorlog.txt

# cat bbb 2> errorlog.txt
# cat errorlog.txt

# wc aaa 2>> errorlog.txt
# cat errorlog.txt
```

## 10.6 Pipes

1. Manipulate and connect the different streams simultaneously.
2. Pipelining
3. Create chain of commands
4. Connects the output of one command to the input of the next command
5. It looks like

```
# command1 | command2 -option | command3 -option1 -option2 | command4
```

```
# ls -l
# ls -l > files.txt
# wc -l files.txt
# ls -l | wc -l
# cd /usr/bin
# ls -l

# man ls
```

# Chapter 11

## Working with Linux Process

### 11.1 What is a process?

1. Anything that is running in Linux is a process
2. Examples
  - (a) Shell that is running and taking our commands
  - (b) The commands that we type on terminal
  - (c) Video in which you are seeing this tutorial
  - (d) Browser in which you have opened the spoken-tutorial.org website
  - (e) Shell scripts that are running
3. A program which is being executed
4. Processes are much like us
  - (a) They are born, they die
  - (b) They have parent and children

### 11.2 Shell process

1. Shell is a process started by Linux Kernel as soon as we login to our system
2. The Linux Kernel is the core of the Linux operating system
3. Consists of the most essential component that make Linux run
4. The shell creates or gives birth to all the other user command processes

\$ date

### 11.3 Spawning

1. A shell can also give birth to another shell process
2. Giving birth to a process or creating a process is also called spawning a process

\$ sh          given birth to new shell  
\$ exit



## 11.4 Process attributes

1. We are identified by attributes like our name, date of birth, etc
2. Similarly processes have attributes
  - (a) PID: Process ID
  - (b) PPID: Parent Process ID
  - (c) Start time, etc
3. PID: Each process is uniquely identified by an unique integer = PID
4. PPID: The PID of the parent of that process

```
$ echo $$      to check the PID of current shell
```

## 11.5 ps command

1. ps(process status) is a command which displays the processes running in the system
2. Let us see what happens if we run this command without any options

```
$ ps
$ sh
$ ps
$ ps -f      more attribute
```

## 11.6 Process types

### 11.6.1 User processes

1. Those processes that are started by the users
2. Eg: ps, most commands that we run on the terminal

### 11.6.2 System processes

1. Those processes that are started
  - (a) by the system often during system startup or
  - (b) user login
  - (c) Eg: bash

## 11.7 All processes

1. To see all the processes
  - (a) System processes
  - (b) User processes
2. We use the -e or the -A option

```
$ ps -e
$ ps -e | more
```

# Chapter 12

## The Linux Environment

### 12.1 Shell Variables

1. Linux can be highly customized by changing the settings of the shell.
2. The behaviour of the shell is generally determined by the shell variables.

#### 12.1.1 Environment Variables

1. Available in user's total environment.
2. Also available in the sub-shells spawned by the shell (like the ones used for running shell scripts)

#### 12.1.2 Local Variables

1. Limited availability
2. Not available in the sub-shells spawned by the shell

\$ set   more	Current Shell Variable
\$ env   more	Environment Variable

```
$ echo $SHELL
$ echo $HOME
$ echo $PATH
$ PATH=$PATH:/home/hemant/
$ echo $LOGNAME          stores user_name of currently logged in user
$ echo $PS2              it gives this'>' prompt changing '$'
$ PS1="@@"
@
@ PS1="$LOGNAME"
hemant
hemant
hemant PS1="$"
$
$                      it is temporary
$
$ history
$ history 10
$ !593
$ !!                    last command
$
$ cd /testtree
$ pwd
$ cd -
$ cd -
$
$
$ alias cdMusic="cd /home/hemant/.../..."
$ cdMusic
$ pwd
/home/hemant/.../...
$ cd -
$ unalias cdMusic
$ cdMusic              error
$
$ alias rm="rm -i"
```

# Chapter 13

## Basics of System Administration

### 13.1 Learning Objectives

1. adduser
2. su
3. usermod
4. userdel
5. id
6. du
7. df

### 13.2 Prerequisite

1. 06-General-Purpose-Utilities
2. One must have admin access in order to execute some admin command

### 13.3 adduser

1. The 'adduser' command will create a new user login for us along with authentication.
2. We can add any user account with the help of 'sudo' command

### 13.4 sudo

1. sudo command allows the administrative user to execute a command as a super user
2. The sudo command has many options

```
$ sudo adduser  
$ sudo adduser duck  
$ ls /home
```

## 13.5 su

1. su stands for 'Switch User'
2. This command is useful in switching from current user to another user

```
$ su - duck
$ logout
```

## 13.6 usermod

1. Enables a super user or root user to modify the settings of other user accounts:
  - (a) Change the password to no password or empty password.
  - (b) Show the date on which the user account will be disabled.

```
$ sudo usermod -e 2012-12-27 duck
```

## 13.7 id

1. id command is used to check the identities of all the "users and groups" on the system.
2. To know about the identity of the user, we use id -u.
3. To know about the identity of the group users, it is id -g.

```
$ id
$ id -u          user id
$ id -n -u      name of user
$ id -g          group id
$ id -G
```

## 13.8 userdel

1. We can delete the user account permanently with the help of the userdel command

```
$ sudo userdel -r duck      user along with home directory
$ ls /home/
```

## 13.9 df and du

1. The df command gives a report on the free space available on the disk.
2. The du command gives a report on how much space a file has occupied

```
$ df
$ df -h          human readable amount

$ cd /home/
$ du -s *.txt
$ du -ch *.txt
```

## 13.10 Summary

1. adduser command to create a new user account.
2. su command to switch from one user to another user.
3. usermod command for changing the user account settings.
4. userdel command to delete the user account.
5. id command to know the information about user ids and group ids.
6. df command to check the file system size and its availability.
7. du command to check the space occupied by a file.

# Chapter 14

## Simple Filters

# sudo su or su root

### 14.1 Prerequisite

Ability to use the mouse, Keyboard, Maximize & Minimize button on a window.

### 14.2 head command

1. head command followed by file name to display first 10 lines of a file by default.
2. Create a file & type these numbers 1, 2, 22, 2, 4, 6, 7, 8, 99, 10, 11, 12
3. Save the file as number.txt

```
$ cat number.txt
$ head number.txt
$ head -n5 number.txt
```

### 14.3 tail command

1. Displays last 10 lines of file by default

```
$ tail number.txt
$ tail -n5 number.txt
```

2. A log file contains events which took place in a system.
3. auth.log file maintains log for who logged in & who logged out.
4. Most useful Option -f

```
$ tail -f /var/log/auth.log
```

### 14.4 Common Linux log files names & Usage

1. /var/log/auth.log: Authentication logs
2. /var/log/message: General & system related stuff
3. /var/log/kern.log: Kernel logs

4. /var/log/cron.log: Crond logs (cron job)
5. /var/log/maillog: Mail server logs
6. /var/log/httpd/: Apache access & error logs
7. /var/log/mysqld.log: MySQL DB server log file

## 14.5 sort command

1. As the name suggest will sort a file in ascending, descending order

```
$ sort number.txt
$ sort -n number.txt
$ sort -r number.txt
$ sort -r -n number.txt
```

2. It can also pull out unique numbers

```
$ sort -un number.txt
```

3. Create a file with the shown contents
4. Let us sort based on the second column

```
$ sort marks.txt -t " " -k2
$ cat marks.txt
```

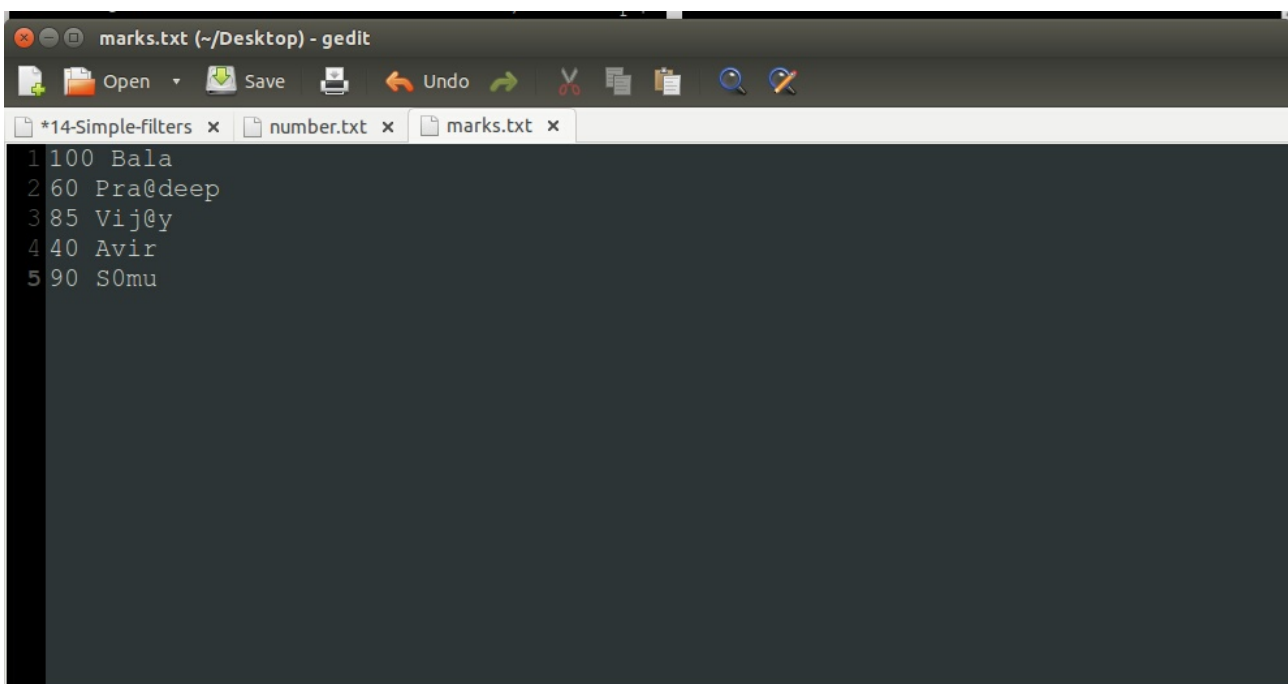


Figure 3: marks.txt

## 14.6 cut command

1. cut is used to cut certain information from a file
2. Let us pull out the names from marks.txt

```
$ cut marks.txt -d " " -f2
```



## 14.7 paste command

1. paste will merge lines of two or more files

```
$ paste number.txt marks.txt
```

2. Redirects it to a new file

```
$ paste number.txt marks.txt > concatfile.txt  
$ cat concatfile.txt
```

3. Print numbers serially

```
$ paste -s number.txt
```

# Chapter 15

## The grep command

```
$ bash -v
```

### 15.1 Regular Expression

1. Regular expressions are pattern matching techniques
2. To kind out whether a pattern exist in a line, paragraph or a file
3. For ex. If you want to search a phone number in the telephone directory  
OR  
To find a keyword in a paragraph or a line, we use grep command

### 15.2 Introduction to grep

1. grep searches for one or more patterns in one or more line, paragraph or a file
2. If filename is not mentioned grep searches for the patterns in standard input
3. If filename is missing, grep searches for the patterns in the standard input

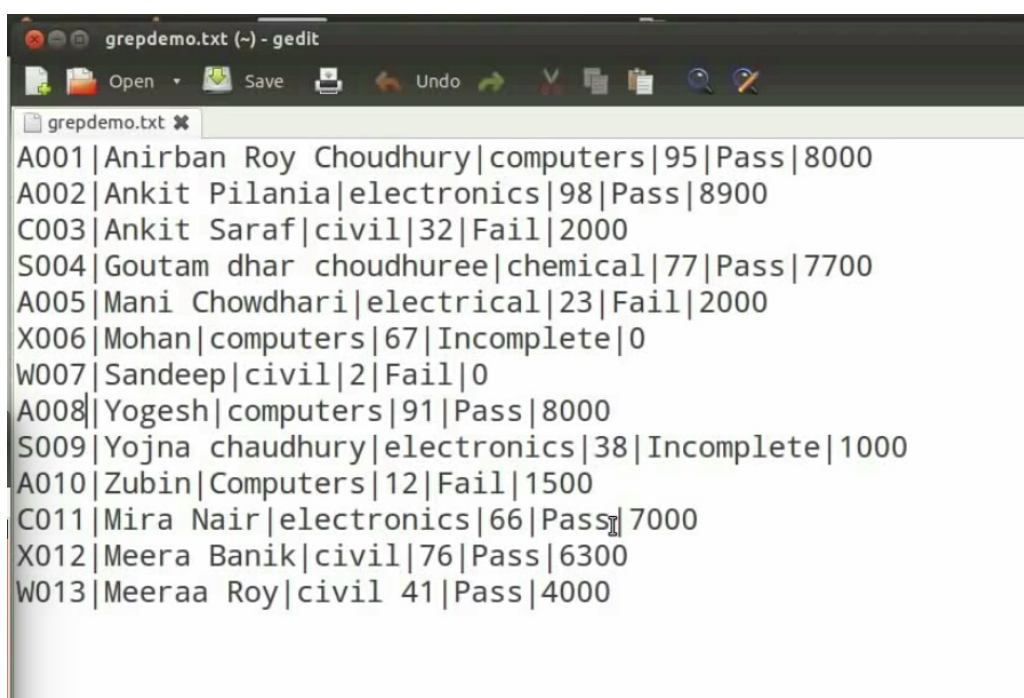


Figure 4: grepdemo.txt

\$ grep "computers" grepdemo.txt	
\$ grep -i "computers" grepdemo.txt	includes C
\$ grep -iv "pass" grepdemo.txt > notpass.txt	v is for not " "
\$ cat notpass.txt	
\$ grep -i "fail" grepdemo.txt	
\$ grep -in "fail" grepdemo.txt	n number
\$ grep -i "ankit saraf" grepdemo.txt	
\$ grep -i "fail" grepdemo.txt notpass.txt	
\$ grep -c "Fail" grepdemo.txt	c count

## 15.3 Summary

1. To see the content of a file

```
$ cat filename
```

2. To list the entries of a particular stream

```
$ grep "computers" grepdemo.txt
```

3. To ignore cases

```
$ grep -i "computers" grepdemo.txt
```

4. Lines that do not match the pattern

```
$ grep -iv "pass" grepdemo.txt
```

5. To list the line numbers with the entries

```
$ grep -in "fail" grepdemo.txt
```

6. To store the result in another file

```
$ grep -iv "pass" grepdemo.txt > notpass.txt
```

7. To know the count

```
$ grep -c "Fail" grepdemo.txt
```

## 15.4 Assignment

-E,+,?

# Chapter 16

## More on grep command

```
$ grep -e "electronics" -e "civil" grepdemo.txt
$ grep -ie "chaudhury" -ie "chowdhari" grepdemo.txt
```

### 16.1 Regular Expressions

1. Provides a concise and flexible means for matching strings of text
2. Such as particular characters, words or patterns of characters
3. There are a number of regular expression characters

### 16.2 The Character Class []

1. Allows us to specify a group of characters within a pair of square brackets
2. Only one out of this group of characters is matched
3. Eg. [abc] would mean that this regular expression matches either a or b or c

```
$ grep -i "ch[ao][uw]dh[ua]r[yi]" grepdemo.txt
```

### 16.3 Range

1. If we want to specify a large range then write:
2. first letter dash last letter of the range
3. Suppose we like to match any digit we simply write [0-9]
4. One out of this group of characters is matched

### 16.4 The Asterisk

1. The \* refers to 0 or more occurrences of the immediately preceding character
2. For example ab\* can match a, ab, abb, abbb etc.

```
$ grep -i "m[ei]*raa*" grepdemo.txt
$ grep "M..." grepdemo.txt
$ grep "Â" grepdemo.txt
$ grep "[78]...$" grepdemo.txt
```

## 16.5 Summary

1. To match more than one pattern

```
$ grep -e "electronics" -e "civil" grepdemo.txt
```

2. To check a word that has different spelling

```
$ grep -ie "choudhury" -ie "chowdhari" grepdemo.txt
```

3. Character class

```
$ grep -i "ch[ao][uw]dh[ua]r[yi]" grepdemo.txt
```

4. The use of Asterisk(\*)

```
$ grep -i "m[ei]*raa*" grepdemo.txt
```

5. To match any one character using dot

```
$ grep "M..." grepdown.txt
```

6. To match a pattern at the beginning of the file

```
$ grep "^A" grepdemo.txt
```

7. To match a pattern at the end of the file

```
$ grep "[78]...$" grepdemo.txt
```

## 16.6 Assignment

List those entries that are 5 letters long and starts with Y

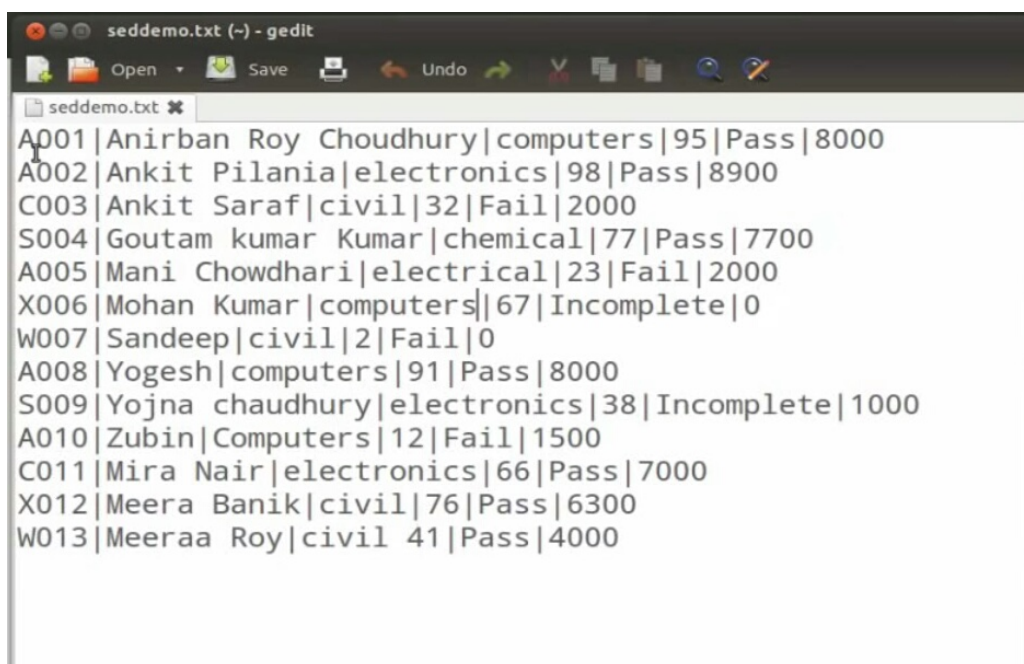
# Chapter 17

## The sed command : The stream editor

### 17.1 Introduction to sed

1. sed is a stream editor
2. sed finds some pattern of text in a particular location of a file
3. It performs some display or editing functions
4. Editing functions like:
  - (a) Insertion
  - (b) Substitution
  - (c) Deletion of matched text

```
$ cat seddemo.txt  
OR  
make a file seddemo.txt
```



```
seddemo.txt (~) - gedit  
seddemo.txt ✖  
A001|Anirban Roy Choudhury|computers|95|Pass|8000  
A002|Ankit Pilania|electronics|98|Pass|8900  
C003|Ankit Saraf|civil|32|Fail|2000  
S004|Goutam kumar Kumar|chemical|77|Pass|7700  
A005|Mani Chowdhari|electrical|23|Fail|2000  
X006|Mohan Kumar|computers|67|Incomplete|0  
W007|Sandeep|civil|2|Fail|0  
A008|Yogesh|computers|91|Pass|8000  
S009|Yojna chaudhury|electronics|38|Incomplete|1000  
A010|Zubin|Computers|12|Fail|1500  
C011|Mira Nair|electronics|66|Pass|7000  
X012|Meera Banik|civil|76|Pass|6300  
W013|Meeraa Roy|civil 41|Pass|4000
```

Figure 5: seddemo.txt

```
$ sed '2p' seddemo.txt
$ sed -n '2p' seddemo.txt      n silent mode, not print unnecessary output
$ sed -n '$p' seddemo.txt      $ last line
$ sed -n '3,6p' seddemo.txt
$ sed -n '3,6!p' seddemo.txt    ! except
```

## 17.2 Line & Context addressing

**Line Addressing** Address specified by the line number

**Context Addressing** Lines that contain particular context say a particular word.

```
$ sed -n '/[cC]omputers/p' seddemo.txt
$ sed -n '/[cC]omputers/w computer_student.txt' seddemo.txt      w save file in filename.txt
$ cat computer_student.txt
$ sed -n -e 'electronic/w electro.txt' -e '/civil/w civil.txt' seddemo.txt
$ cat electro.txt
$ cat civil.txt
```

## 17.3 Summary

1. sed
2. To print using sed
3. Line Addressing
4. Context Addressing

## 17.4 Assignment

1. Use the same text file seddemo.txt
2. Try to print records from 6th to 12th line

# Chapter 18

## More on sed command

The major use of sed is substitution

Replacing some pattern in the input with something else

### 18.1 Substitution & Replacement

```
$ sed 's/[kK]umar/Roy/' seddemo.txt
$ sed 's/[kK]umar/Roy/g' seddemo.txt          g everywhere
$ sed -e 's/electronics/electrical/g' -e 's/civil/metallurgy/g' seddemo.txt
$ sed '/Anirban/s/computers/mathematics/g' seddemo.txt
$ sed '/electronics/d' seddemo.txt > nonelectronics.txt
$ cat nonelectronics.txt
```

### 18.2 Insertion

```
$ sed '1i Student Information' seddemo.txt
$ sed '1i Student Information\n2013' seddemo.txt
```

### 18.3 Summary

1. Substitution
2. Replacement
3. Insertion

### 18.4 Assignment

1. Use the same text file seddemo.txt
2. Try to replace or substitute name Ankit with Ashish



# Chapter 19

## Basics of awk

### 19.1 Introduction

1. awk is a very powerful text manipulation tool
2. It is named after its authors Aho, Weinberger and Kernighan
3. It can perform several functions
4. It operates at the field level of a record
5. So, it can easily access and edit the individual fields of the record

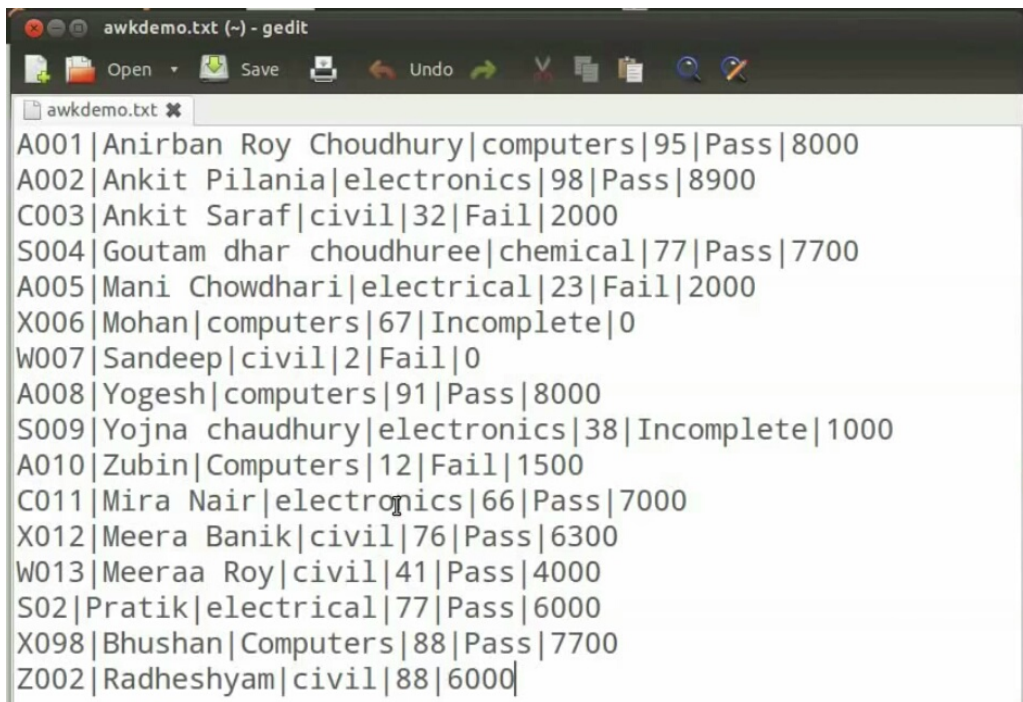


Figure 6: awkdemo.txt

```
$ awk '/Pass/print' awkdemo.txt
$ awk '/M[ei]*ra*/print' awkdemo.txt
$ awk '/civil|electrical/ print' awkdemo.txt
```

## 19.2 Parameters

1. awk has some special parameters to identify individual fields of a line
2. \$1(Dollar 1) would indicate the first field
3. Similarly we can have 2,3 and so on, for respective fields
4. \$0 represents the entire line
5. | delimiter

```
$ awk -F "|" '/civil|electrical/print $0' awkdemo.txt
$ awk -F "|" '/civil|electrical/print $2,$3' awkdemo.txt
$ awk -F "|" '/Pass/printf"%4d %-25s %-15s \n",NR,$2,$3' awkdemo.txt
```

## 19.3 Summary

1. To print using awk
2. Regular expression in awk
3. To list the entries for a particular stream
4. To list only the second and the third fields
5. To display a formatted output

## 19.4 Assignment

Display roll no., stream and marks of Ankit Saraf