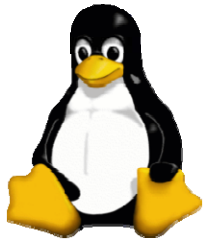
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE

CSN-101 (Introduction to Computer Science and Engineering)

Lecture 12: Linux Operating System and Linux Administration



Dr. Sudip Roy

Assistant Professor

Department of Computer Science and Engineering

Piazza Class Room: https://piazza.com/iitr.ac.in/fall2019/csn101

[Access Code: csn101@2019]

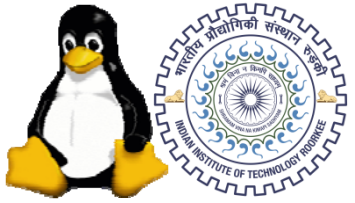
Moodle Submission Site: https://moodle.iitr.ac.in/course/view.php?id=45

[Enrollment Key: csn101@2019]

Plan for Lecture Classes in CSN-101 (Autumn, 2019-2020)

Week	Lecture 1 (Monday 4-5 PM)	Lecture 2 (Friday 5-6 PM)	
1	Evolution of Computer Hardware and Moore's Law, Software and Hardware in a Computer	Computer Structure and Components, Operating Systems	MTE
2	Computer Hardware: Block Diagrams, List of Components	Computer Hardware: List of Components, Working Principles in Brief, Organization of a Computer System	
3	Linux OS	Linux OS	
4	Writing Pseudo-codes for Algorithms to Solve Computational Problems	Writing Pseudo-codes for Algorithms to Solve Computational Problems	ETE
5	Sorting Algorithms – Bubble sort, selection sort, and Search Algorithms	Sorting Algorithms – Bubble sort, selection sort, and Search Algorithms	
6	C Programming	C Programming	
7	Number Systems: Binary, Octal, Hexadecimal, Conversions among them	Number Systems: Binary, Octal, Hexadecimal, Conversions among them	
8	Number Systems: Negative number representation, Fractional (Real) number representation	Boolean Logic: Boolean Logic Basics, De Morgan's Theorem, Logic Gates: AND, OR, NOT, NOR, NAND, XOR, XNOR, Truth-tables	
9	Computer Networking and Web Technologies: Basic concepts of networking, bandwidth, throughput	Computer Networking and Web Technologies: Basic concepts of networking, bandwidth, throughput	MTE
10	Different layers of networking, Network components, Type of networks	Network topologies, MAC, IP Addresses, DNS, URL	
11	Different fields of CSE: Computer Architecture and Chip Design	Different fields of CSE: Data Structures, Algorithms and Programming Languages	ETE
12	Different fields of CSE: Database management	Different fields of CSE: Operating systems and System softwares	
13	Different fields of CSE: Computer Networking, HPCs, Web technologies	Different Applications of CSE: Image Processing, CV, ML, DL	Term Project
14	Different Applications of CSE: Data mining, Computaional Geometry, Cryptography, Information Security	Different Applications of CSE: Cyber-physical systems and IoTs	

System Administration command in Linux



chmod – change file access permissions

Syntax: chmod [OPTION] [MODE] [FILE]

e.g., chmod 744 calculate.sh

chown – change file owner and group

Syntax: chown [OPTION]... OWNER[:[GROUP]] FILE...

e.g., chown remo myfile.txt

su – change user ID or become superuser

Syntax: su [OPTION] [LOGIN]

e.g., su remo, su

passwd – update a user's authentication tokens(s)

Syntax: passwd [OPTION]

e.g., Passwd

who – show who is logged on

Syntax: who [OPTION]

e.g., who , who b , who q

Manual and online help for commands

Man (Manual)

- **man** is the interface used to view the system's reference manuals.

Syntax:

man [option(s)] keyword(s)

- But generally [option(s)] are not used. Only keyword is written as an argument.

Example

- man ls

man Options

Commands	Function
man -aw	List all available sections of a command.
man -a	To view all man pages of a command.
sman -k (apropos)	Shows a list of results in man page containing a keyword match.
-f, whatis	It displays description from manual page if available.
whereis	Used to determine location of a man page

help

Help:

- Display information about built-in commands.

Syntax

help [-dms] [PATTERN ...]

Options

- -d Output short description for each topic.
- -m Display usage in pseudo-manpage format.
- -s Output only a short usage synopsis for each topic matching.

General Purpose Utilities

The **general-purpose utilities** of the system : divided into two categories:

- Some commands tell you the state of the system.
- Others can aid you directly in your work.

- cal - Displays a calendar
- date - print or set the system date and time
- bc - An arbitrary precision calculator language
- echo - Display a line of text.
- printf - Format and print data
- passwd - update user's authentication tokens
- who - Show who is logged on
- w - Show who is logged on and what they are doing
- uname - Print system information
- expr - Evaluate expressions
- test - Check file types and compare values
- seq - Print a sequence of numbers
- factor - Factor numbers
- rev - reverse lines of a file or files

File System

- All data in Unix is organized into files.
- All files are organized into directories.
- Directories are organized into a tree-like structure called the filesystem

There are three basic types of files –

- **Ordinary Files** – An ordinary file is a file on the system that contains data, text, or program instructions. In this tutorial, you look at working with ordinary files.
- **Directories** – Directories store both special and ordinary files. For users familiar with Windows or Mac OS, Unix directories are equivalent to folders.
- **Special Files** – Some special files provide access to hardware such as hard drives, CD-ROM drives, modems, and Ethernet adapters. Other special files are similar to aliases or shortcuts and enable you to access a single file using different names.

Creating and managing files & directories

Listing Files

- To list the files and directories stored in the current directory, use the following command: $ls

Creating Files

- You can use the vi editor to create ordinary files on any Unix system. You simply need to give the following command: $ vi filename

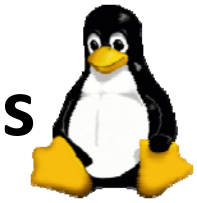
Editing Files

You can edit an existing file using the vi editor. We will discuss in short how to open an existing file: $ vi filename

Display Content of a File

- You can use the cat command to see the content of a file. Following is a simple example to see the content of the above created file: $ cat filename

Creating and managing files & directories



Counting Words in a File

- You can use the wc command to get a count of the total number of lines, words, and characters contained in a file. Following is a simple example to see the information about the file created above: $ wc filename

Copying Files

To make a copy of a file use the cp command. The basic syntax of the command is –

$ cp source_file destination_file

Renaming Files

To change the name of a file, use the mv command. Following is the basic syntax –

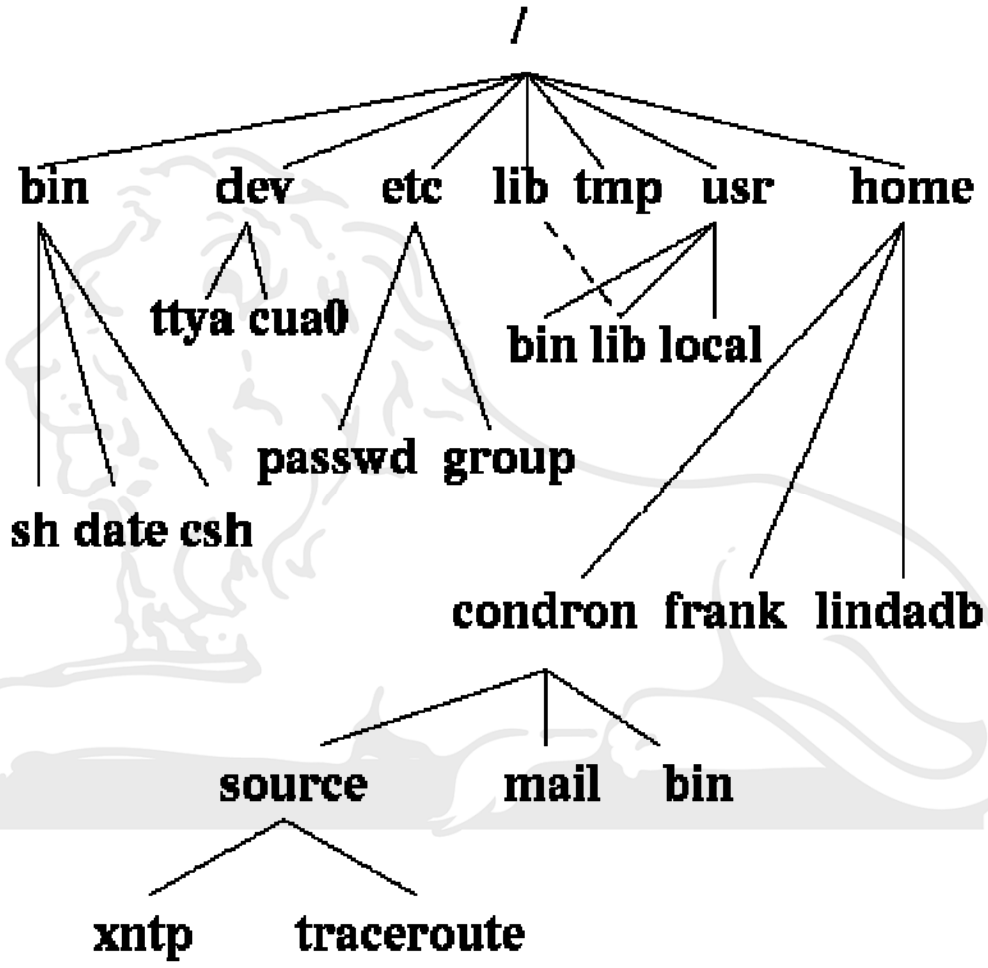
$ mv old_file new_file

Deleting Files

To delete an existing file, use the rm command. Following is the basic syntax –

$ rm filename

UNIX File System



Linux Filesystem Hierarchy Standard (FHS)

- Filesystem hierarchy standard describes directory structure and its content in Unix and Unix like operating system.

- Explains where files and directories should be located and what it should contain.

- **The Root Directory:** All the directories in the Linux system comes under the root directory which is represented by a forward slash (/). Everything in your system can be found under this root directory even if they are stored in different virtual or physical devices.

Linux Directories

Categorize the directories according to the type of file:

Directory type	Types of files stored
Binary directories	Contains binary or compiled source code files, e.g., /bin, /sbin, etc.
Configuration directories	Contains configuration files of the system, e.g., /etc, /boot.
Data directories	Stores data files, e.g., /home, /root, etc.
Memory directories	Stores device files which doesn't take up actual hard disk space, e.g., /dev, /proc, /sys.
Usr (Unix System Resources)	Contains sharable, read only data, e.g.., /usr/bin, /usr/lib, etc.
var (variable directory)	Contains larger size data, e.g., /var/log, /var/cache, etc.
Non-standard directories	Directories which do not come under standard FHS, e.g., lost+found, /run, etc.

Linux Directories

Examples of Subdirectories

/root	directory, starting point of the directory tree
/home	(private) directories of users
/devDevice	files that represent hardware components
/etc	Important files for system configuration
/etc/init.d	Boot scripts/usr/binGenerally accessible programs

Linux Binary Directory

Binary files are the files which contain compiled source code (or machine code). They are also called **executable files** because they can be executed on the computer.

Binary directory contains following directories:
/bin
/sbin
/lib
/opt

/bin' directory: contains user binaries, executable files, Linux commands that are used in single user mode, and common commands that are used by all the users, like cat, cp, cd, ls, etc.

The '/bin' directory doesn't contain directories.

Example:

ls /bin

Linux Binary Directory

/sbin

- The '/sbin' directory also contains executable files, but unlike '/bin' it only contains system binaries which require root privilege to perform certain tasks and are helpful for system maintenance purpose.
- e.g., fsck, root, init, ifconfig, etc.

Example:

ls /sbin

/lib

- The '/lib' directory contains shared libraries which are often used by the '/bin' and '/sbin' directories. It also contains kernel module. These filenames are identable as ld* or lib*.so.*. For example, ld-linux.so.2 and libfuse.so.2.8.6

Example:

- ls /lib

/opt

- The term 'opt' is short for optional. Its main purpose is to store optional application software packages.
- Add-on applications from individual vendors should be installed in '/opt'

File and Directory Operations

- **cat -** concatenate files and print on the standard output
- **touch -** change file timestamps
- **cp -** copy files and directories
- **rm -** remove files or directories
- **mv -** move (rename) files
- **more -** file perusal filter for crt viewing
- **file -** determine file type
- **wc -** print newline, word, and byte counts for each file
- **od -** dump files in octal and other formats
- **cmp -** compare two files
- **comm -** compare two sorted files line by line
- **diff -** find differences between two files

Unix / Linux – File Permission / Access Modes

Every file in Linux/Unix has the following attributes −

- **Owner permissions** − The owner's permissions determine what actions the owner of the file can perform on the file.
- **Group permissions** − The group's permissions determine what actions a user, who is a member of the group that a file belongs to, can perform on the file.
- **Other (world) permissions** − The permissions for others indicate what action all other users can perform on the file.

Changing File Ownership

Using 2 commands we can change file and directory ownership:

1. chown command
2. chgrp command

CHOWN COMMAND:

- used to change the file ownership.
- This command transfers ownership of a file to user.

Syntax

- $ chown options owner [:group] file(s)

CHGRP COMMAND:

- This command is used to change the file's group owner.

Syntax

- $ chgrp options owner [:group] file(s)

File / Directory Access Modes

File Access Modes

The permissions of a file are the first line of defense in the security of a Unix system. The basic building blocks of Unix permissions are the read, write, and execute permissions, which have been described below:

- **Read:** Grants the capability to read, i.e., view the contents of the file.
- **Write:** Grants the capability to modify, or remove the content of the file.
- **Execute:** User with execute permissions can run a file as a program.

Directory Access Modes

Directory access modes are listed and organized in the same manner as any other file. There are a few differences that need to be mentioned:

- **Read:** Access to a directory means that the user can read the contents. The user can look at the filenames inside the directory.
- **Write:** Access means that the user can add or delete files from the directory.
- **Execute:** Executing a directory doesn't really make sense, so think of this as a traverse permission.

Access Permission of File/Directory

- The #'s can be:
- 0 = Nothing
  1 = Execute
  2 = Write
  3 = Execute & Write (2 + 1)
  4 = Read
  5 = Execute & Read (4 + 1)
  6 = Read & Write (4 + 2)
  7 = Execute & Read & Write (4 + 2 + 1)

Handling Ordinary Files

cat

- To display file content
- The 'cat' command can be used to display the content of a file.

Syntax:

cat **<fileName>**

Some other commands:

- pwd (print [current] working directory) pwd displays the full absolute path to the your current location in the filesystem. ...
- cd (change [current working] directory) $ cd path. ...
- ls (list directory) ...
- mkdir (make directory) ...
- rmdir (remove directory) ...
- cp (copy) ...
- mv (move/rename) ...
- rm (remove/delete)

grep

- The grep command will search for line that matches the specified pattern.
- Print Lines Matching A Pattern

The grep tool has the following options to use regular expressions:

- -E : String is read as ERE (Extended Regular Expressions)
- -G : String is read as BRE (Basic Regular Expressions)
- -P : String is read as PRCE (Perl Regular Expressions)
- -F : String is read literally.

Syntax:

grep **<pattern> <fileName>**

Concatenating Characters

If a pattern is of concatenating characters then it has to be matched as it is, for the line to be displayed.

grep

One or the Other

Here pipe (|) symbol is used as OR to signify one or the other.

Three versions are shown.

Options -E and -P syntax are same but -G syntax uses (\).

Syntax:

grep <option> <'pattern|pattern> <fileName>

One or More / Zero or More

- The * signifies zero or more times occurrence of a pattern and + signifies one or more times occurrence.

Syntax:

grep <option> <'pattern*'> <fileName>

Match the End of a String

- To match the end of a string we use $ sign.

Syntax:

grep <pattern>$ <fileName>

vi editor

- The vi editor is elaborated as visual editor.
- It is installed in every Unix system.
- It is a very powerful application. An improved version of vi editor is vim but most Linux systems have vi editor installed.

vi editor has two modes:

- **Command Mode**: In command mode, actions are taken on the file. The vi editor starts in command mode. Typed words will act as commands in vi editor. To pass a command you have to be in command mode.

- **Insert Mode:** In insert mode, entered text will be inserted into the file. Esc key will take you to the command mode from insert mode.

Key Points:

- By default, vi editor starts in command mode. To enter text, you have to be in insert mode, just type 'i' and you'll be in insert mode. Although, after typing i nothing will appear on the screen but you'll be in insert mode.
- To exit from insert mode press Esc key, you'll be directed to command mode.

vi editor

To save and quit

- You can save and quit vi editor from command mode. Before writing save or quit command you have to press colon (:). Colon allows you to give instructions to vi.

Commands	Action
:wq	Save and quit
:w	Save
:q	Quit
:w fname	Save as fname
ZZ	Save and quit
:q!	Quit discarding changes made
:w!	Save (and write to non-writable file)

exit vi table

vi editor

- Type :wq to **save and exit the file**
- If you want to **quit without saving** the file, use :q. This command will only work when you have not made any changes in the file.
- command :!q It discards the changes made in the file and save it.

vi Commands

Linux vi editor is different from other editors. You have to use different keys to use different functions. Although, it's quite easy and interesting to use vi editor.

The vi editor commands are case sensitive.

To switch from command to insert mode:

Command	Action
i	Start typing before the current character
I	Start typing at the start of current line
a	Start typing after the current character
A	Start typing at the end of current line
o	Start typing on a new line after the current line
O	Start typing on a new line before the current line

vi Commands

To move around a file:

Commands	Action
j	To move down
k	To move up
h	To move left
l	To move right

To Jump Lines:

Commands	Action
G	Will direct you at the last line of the file
``	Will direct you to your last position in the file

vi Commands

To Delete:

Commands	Action
x	Delete the current character
X	Delete the character before the cursor
r	Replace the current character
xp	Switch two characters
dd	Delete the current line
D	Delete the current line from current character to the end of the line
dG	delete from the current line to the end of the file

vi Commands

To repeat and undo:

Commands	Action
u	Undo the last command
.	Repeat the last command

Command to cut, copy and paste:

Commands	Action
dd	Delete a line
yy	(yank yank) copy a line
p	Paste after the current line
P	Paste before the current line

Command to cut, copy and paste in blocks:

Commands	Action
<n>dd	Delete the specified n number of lines
<n>yy	Copy the specified n number of lines

vi Commands

Start and end of line:

Commands	Action
θ	Bring at the start of the current line
^	Bring at the start of the current line
$	Bring at the end of the current line
dθ	Delete till start of a line
d$	Delete till end of a line

Joining lines:

Commands	Action
J	Join two lines
yyp	Repeat the current line
ddp	Swap two lines

vi Commands

Move forward or backward:

Commands	Action
w	Move one word forward
b	Move one word backward
<n>w	Move specified number of words forward
dw	Delete one word
yw	Copy one word
<n>dw	Delete specified number of words

Linux Shell

- A Shell provides you with an interface to the Unix system.
- It gathers input from you and executes programs based on that input. When a program finishes executing, it displays that program's output.
- Shell is an environment in which we can run our commands, programs, and shell scripts When you log into the system you are given a default shell.

Shell Prompt

- The prompt, $, which is called the command prompt, is issued by the shell. While the prompt is displayed, you can type a command.
- Shell reads your input after you press Enter. It determines the command you want executed by looking at the first word of your input.

Shell Types

In Linux, there are two major types of shells –

- Bourne shell – If you are using a Bourne-type shell, the $ character is the default prompt.
- C shell – If you are using a C-type shell, the % character is the default prompt.

The Bourne Shell has the following subcategories –

- Bourne shell (sh)
- Korn shell (ksh)
- Bourne Again shell (bash)
- POSIX shell (sh)

The different C-type shells follow –

- C shell (csh)
- TENEX/TOPS C shell (tcsh)

Shell Scripts

- The basic concept of a shell script is a list of commands, which are listed in the order of execution.
- shell script will have comments, preceded by # sign, describing the steps.
- There are conditional tests, such as value A is greater than value B, loops allowing us to go through massive amounts of data, files to read and store data, and variables to read and store data, and the script may include functions.
- Shell scripts and functions are both interpreted. This means they are not compiled.

Shell Programming

- You can write shell programs by creating scripts containing a series of shell commands.
- The first line of the script should start with #! which indicates to the kernel that the script is directly executable.
- You immediately follow this with the name of the shell, or program (spaces are allowed), to execute, using the full path name. So to set up a Bourne shell script the first line would be: #! /bin/sh

Shell variables

Variable Names

The name of a variable can contain only letters (a to z or A to Z), numbers (0 to 9) or the underscore character (_).

Defining Variables

Variables are defined as follows –
variable_name=variable_value

Accessing Values

To access the value stored in a variable, prefix its name with the dollar sign ($) –

Read-only Variables

Shell provides a way to mark variables as read-only by using the read-only command. After a variable is marked read-only, its value cannot be changed.

Unsetting Variables

Unsetting or deleting a variable directs the shell to remove the variable from the list of variables that it tracks. Once you unset a variable, you cannot access the stored value in the variable.

Syntax:

unset variable_name

Variable Types

When a shell is running, three main types of variables are present:

Local Variables – A local variable is a variable that is present within the current instance of the shell. It is not available to programs that are started by the shell.
They are set at the command prompt.

Environment Variables – An environment variable is available to any child process of the shell. Some programs need environment variables in order to function correctly.

Shell Variables – A shell variable is a special variable that is set by the shell and is required by the shell in order to function correctly. Some of these variables are environment variables whereas others are local variables.

Some commonly used ENVs in Linux

- $USER: Gives current user's name.
- $PATH: Gives search path for commands.
- $PWD: Gives the path of present working directory.
- $HOME: Gives path of home directory.
- $HOSTNAME: Gives name of the host.
- $LANG: Gives the default system language.
- $EDITOR: Gives default file editor.
- $UID: Gives user ID of current user.
- $SHELL: Gives location of current user's shell program.

Linux Shell Commands

type command

Linux 'type' command tell us whether a command given to the shell is a built-in or external command.

Syntax:
type <command>

Example:
type pwd
type cd

type -a

The 'type -a' option tells about all type of command whether it is built-in, external, or aliased. Some commands are both external and built-in commands.
Syntax:
type -a <command>

Example:
type -a echo

Linux Shell Commands

which

Linux 'which' command locates the path of a command.

Syntax:

which <command1> <command2> <command3>....

Show All Running Processes in Linux

ps command: It provide information about the currently running processes, including their process identification numbers (PIDs).

- If you want a repetitive update of this status, use top, atop, and/or htop command as described below.

Type the following ps command to display all running process:
ps -aux | less

OR
ps aux | less

Where,

A : Select all processes
u : Select all processes on a terminal, including those of other users
x : Select processes without controlling ttys

Show All Running Processes in Linux

top command

The top program provides a dynamic real-time view of a running system.
Syntax:
top

Display a tree of processes

pstree shows running processes as a tree. The tree is rooted at either pid or init if pid is omitted. If a user name is specified, all process trees rooted at processes owned by that user are shown.
Syntax:
$ pstree

Lookup process

Use pgrep command. pgrep looks through the currently running processes and lists the process IDs which matches the selection criteria to screen.

Shell Scripting

- Shell Scripting is an open-source operating system.

- In Linux, shells like bash and korn support programming construct which are saved as scripts. These scripts become shell commands and hence many Linux commands are script.

How to determine Shell

You can get the name of your shell prompt, with following command:

Syntax:

```
echo $SHELL
```

Linux vs Windows

- Financial Differences
- Technical Differences
- End-User Differences

Linux vs Windows

- Financial Differences

COST		
	LINUX	WINDOWS
Online Downloads	Free	Not Available
Retail Price, CD	$50	$300

Linux vs Windows

Technical Differences

- Keeping up to date
  - By Upgrading
    - Linux upgrades faster than Windows
- Compatibility
  - Linux is Backward Compatible unlike Windows

Features Provided

- Both support Dynamic Caching
- Both have Multi-user Support

Linux vs Windows

End-User Differences

- Proprietary vs. Open Source
  - Windows is a Proprietary Technology

    Applications will only work on Windows
  - Linux – Open Source

Linux vs Windows

In the Commercial Arena

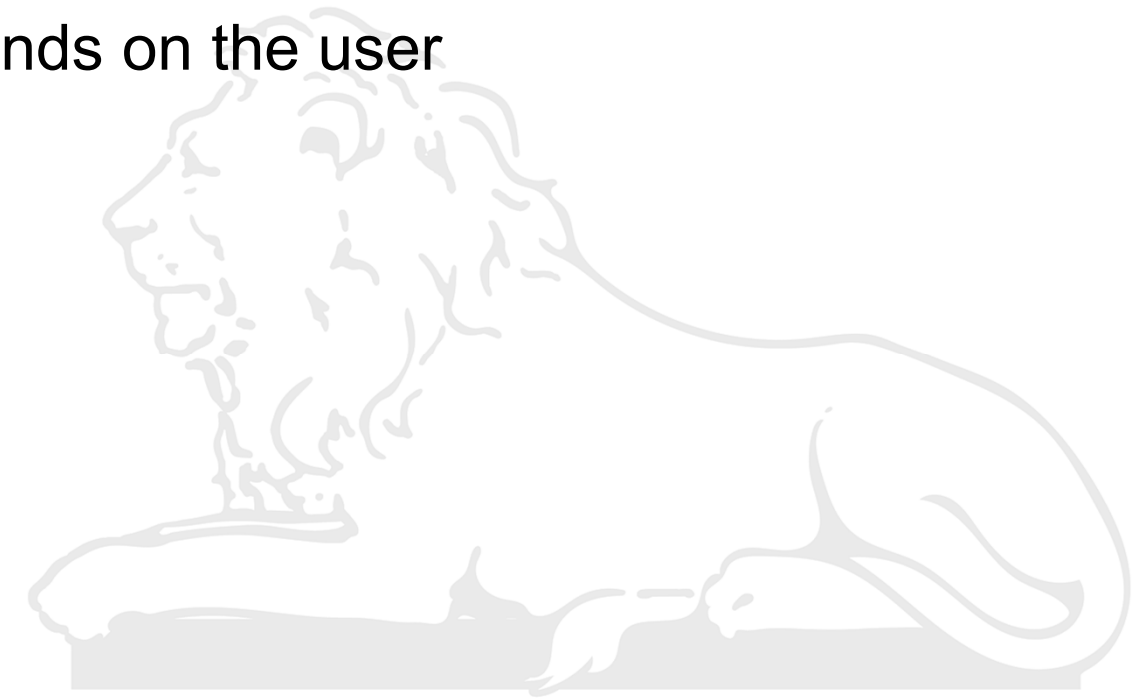
- Head to head competition
- Used side by side as servers
- Both handled daily workload for several small business operations
- Linux with hardware disadvantage supported a community of users 3 times size of NT's

- NT – easy for non-programmer
- Linux – programmer-based culture

Conclusions:

"When is it best to use Linux and when should some other operating system be preferred?"

- It all depends on the user



Homework Assignments:

- Login as guest (password is guest)
- Find the present Directory
- Write the root directory structure
- Write a few commands available in /bin and /sbin directory
- Find the guest directory
- Write the permissions of guest directory
- Create a new Directory test in guest directory
- Copy the file /etc/resolv.conf in test directory
- Rename the test directory to testing
- Delete the testing directory
- Change the permissions of guest directory to 700
- Change the permissions of /tmp directory to 700