



Mu Sigma

Introduction to Linux

Do The Math

Chicago, IL
Bangalore, India
www.mu-sigma.com

November, 2013

Proprietary Information

"This document and its attachments are confidential. Any unauthorized copying, disclosure or distribution of the material is strictly forbidden"



Agenda

- ▶ Introduction to Linux
- ▶ Linux Architecture and Distributions
- ▶ Linux File System
- ▶ Useful Linux Commands
 - Exploring the File System
 - Text Editors and File viewing
 - Permissions and Ownership
 - Process Management
 - Installing and Uninstalling Software
 - File Compression
 - Networking
 - Miscellaneous
- ▶ Environment Variables
- ▶ Shell Script Basics
- ▶ Exercises

What is Linux?

- ▶ Linux is a common abbreviation of **GNU/Linux** which is a **FREE OF COST** replacement for proprietary operating system built under the model of free software development and distribution. It comprises the GNU System developed by the GNU Project and Linux kernel
- ▶ Linux is easy to use and robust
 - Linux is a fast operating system with minimum hardware requirements and also has a modern Graphical User Interface which can be tailored to user needs.
 - Linux contains thousands of ready to use applications and utilities
 - It lets one to customize your desktop to suit your exact needs and create your own personal perfect user interface

“ Our most potent Operating System competitor is Linux and the phenomena around Open Source and free software. The same phenomena fuels competitors to all of our products. The ease of picking up Linux to learn it or to modify some piece of it is very attractive. The academic community, start up companies, foreign governments and many other constituencies are putting their best work into Linux.”

Bill Gates, Microsoft CEO, 2001



Image courtesy: www.sudoroom.org

Linux – Brief History

- ▶ More than 20 years ago, Linus Torvalds sparked an open source revolution with a short email declaring he was doing a new project “just for fun.”
- ▶ Today, Linux powers 98% of the world’s super computers, most of the servers powering the Internet, the majority of financial trades worldwide and tens of millions of Android mobile phones and consumer devices.

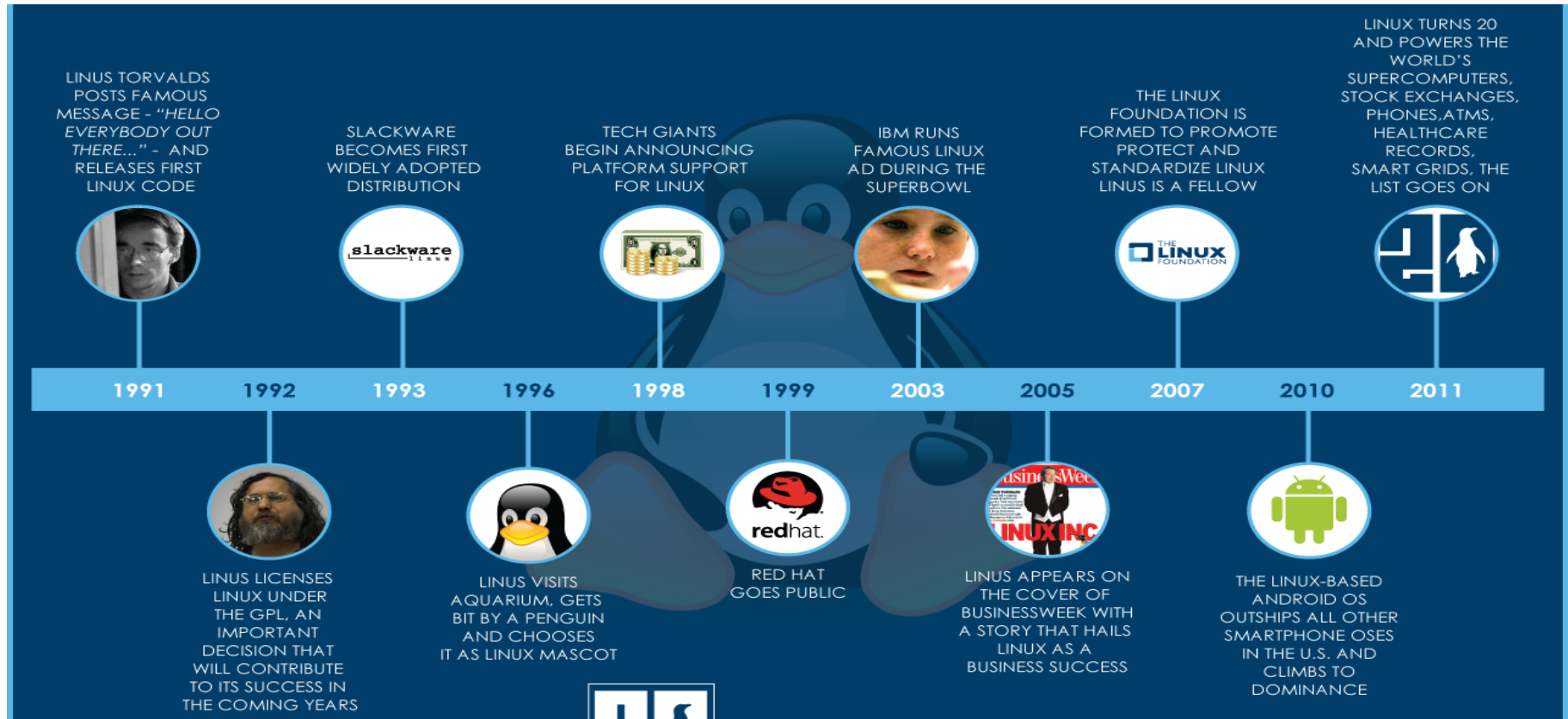


Image Courtesy: www.linuxfoundation.org

Why Linux?

▶ Independence & Free of Cost

- Linux is under GNU General Public Licence (GPL), this license gives a right to copy the software, look at the source code, modify and redistribute it to anyone.
- This includes unlimited, lifetime, free of cost updates and security patches.

▶ Support & Security

- Linux was built by users for users. There are a huge number of online forums available to provide you with 24/7 support.
- Provides better security as it was built on top of a powerful authorization management system, which does not allow regular users to interact the core system functions hence reducing virus attacks.

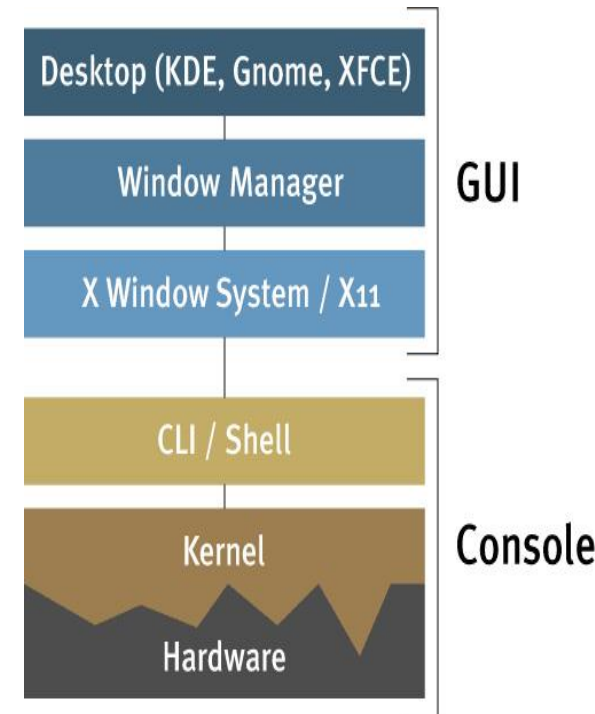
▶ Performance & Reliability

- Most of the world's modern supercomputers use Linux due to its profound performance and reliability.
- Linux is a flexible operating system supporting multiple hardware platforms and yet limiting hardware requirements to run this operating system to a bare minimum.



Basic Architecture

- ▶ **Kernel**- It acts as an interface between the operating system and other hardware resources. It is the bridge between the application and the actual data processing.
- ▶ **Shell**- It acts as an interface between user and the operating system. It is the software that provides an interface for user of an operation system which need services of the kernel.
- ▶ **System Utilities**- The System Utilities consists of various system interrupts and system calls which is to transfer the control from the user mode to the kernel mode containing the kernel and shell for further execution of the commands.
- ▶ **GUI**- A window manager is system software that controls the placement and appearance of windows within a windowing system in a graphical user interface. Various desktop environments are discussed further



Linux Desktop Environments

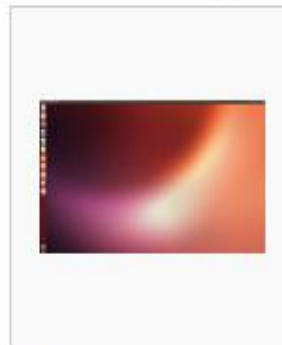
- ▶ The selection of top Linux GUIs differs based on the purpose – i.e. if you will use Linux on a server, the choice is different from that when you are choosing a Linux desktop



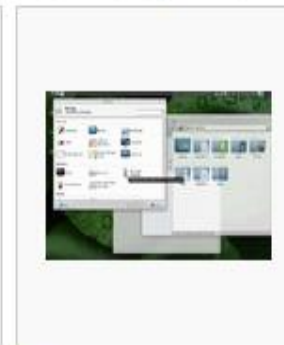
GNOME Shell (GNOME 3)



KDE Plasma (KDE 4)



Unity



Xfce



LXDE



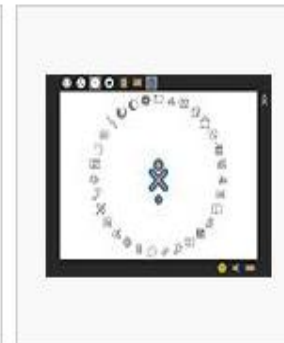
Cinnamon



MATE (GNOME 2)



Trinity (KDE 3)



Sugar



Pantheon

Image Courtesy: www.wikipedia.org

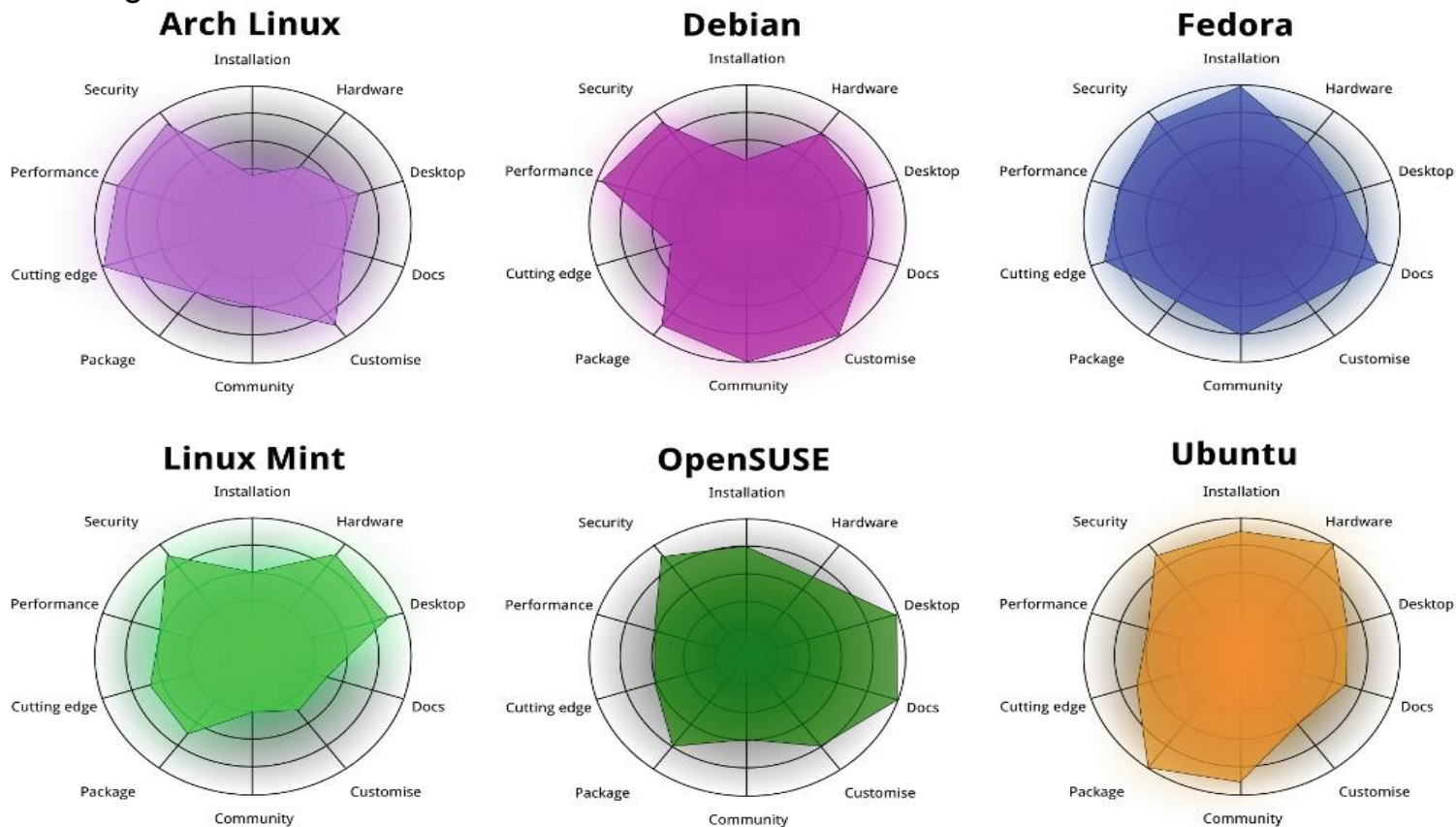
Types of Shells

Shells are command-line interfaces which provide powerful environments for software development and system maintenance. Though shells have many commands in common, each type has unique features.

- ▶ **sh**- The Bourne shell, called "sh," is one of the original shells, It offers features such as input and output redirection, shell scripting with string and integer variables, and condition testing and looping.
- ▶ **bash**- Linux systems still offer the sh shell, but "bash" -- the "Bourne-again Shell," based on sh -- has become the new default standard
- ▶ **cs**h and **tc**sh- Using C syntax as a model the "C-shell", csh was developed. Further tcsh fixed problems in csh and added command completion.
- ▶ **k**sh- ksh improves on the Bourne shell by adding floating-point arithmetic, job control, command aliasing and command completion

Which Linux distribution to use?

- ▶ Linux development has already produced hundreds of Linux distributions for end-users.
- ▶ A wide variety of products makes it hard to choose the right one. Many Linux users try out different versions before deciding on a Linux distribution.



Installing Linux

- ▶ Each distribution has its own installation and maintenance utilities that ease installation and system administration. Each is aimed at a different audience.
- ▶ Linux gives you freedom to choose. Be it purchasing a CD-ROM or simply borrowing a friend's copy or downloading the source from the Internet. <http://distrowatch.com/> lets one choose and download any version.
- ▶ Whether purchased from a major Linux distributor or downloaded from their FTP site, you get the same operating system and the software packages that they offer.



Remote system access

▸ ssh

SSH, which is an acronym for Secure SHell, was designed and created to provide the best security when accessing another computer remotely.

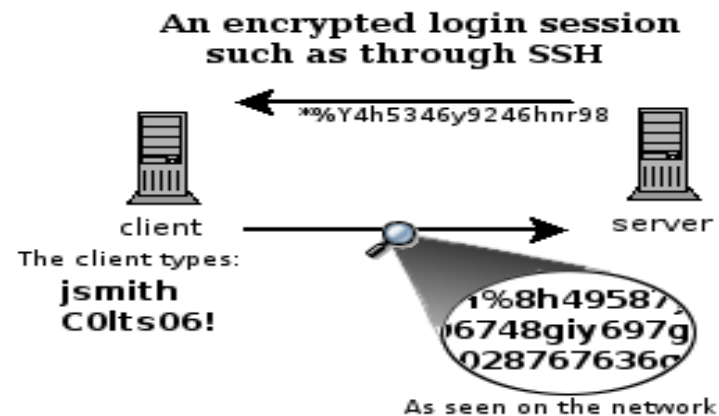


Fig.9

Syntax	Example
<code>ssh user@hostname [command]</code>	<code>\$ssh nihit@172.25.2.180</code>
	<code>\$ssh jhawkins@collie.stanford.edu uptime</code>

Ending an SSH can be done using exit, logout commands or Ctrl+D shortcut.

File Transfer

▸ scp

It is basically a program that uses the SSH protocol to send files between hosts over an encrypted connection.

You can transfer files from your local computer to a remote host or vice versa or even from a remote host to another remote host.

Syntax	Example
<pre>scp [-r] user@host1:file1 user@host2:file2</pre>	<pre>\$scp xyz@172.54.1.91:xyz.txt abc@172.54.1.160:~</pre>

▸ ftp

FTP is a standard network protocol used to transfer files from one host to another host over a TCP-based network, such as the Internet. Clients like Filezilla, Cyberduck, WinSCP are the [popular ones](#).

Syntax	Example
<pre>ftp [-d][-g][-i][-n][-v] [HostName [Port]]</pre>	<pre>#To connect your local machine to the remote machine, type \$ftp 123.23.1.123 #When you enter your own <i>loginname</i> and <i>password</i> #for the #remote machine, it returns the prompt ftp></pre>

Logging into LINUX File System

Note:ssh server must be installed and running for you to log into a linux system

Mac or LINUX

- ▶ To log-in into the remote Linux shell, open terminal and type:

```
ssh -X <your_username>@<host_name>
```

host name is the remote server's domain name or IP address (e.g. cluster.ucr.edu)

You will be asked to enter the password, simply type it and press enter.

- To copy files **To** the server run the following on your workstation or laptop:

```
scp -r <path_to_directory> <your_username>@<host_name>
```

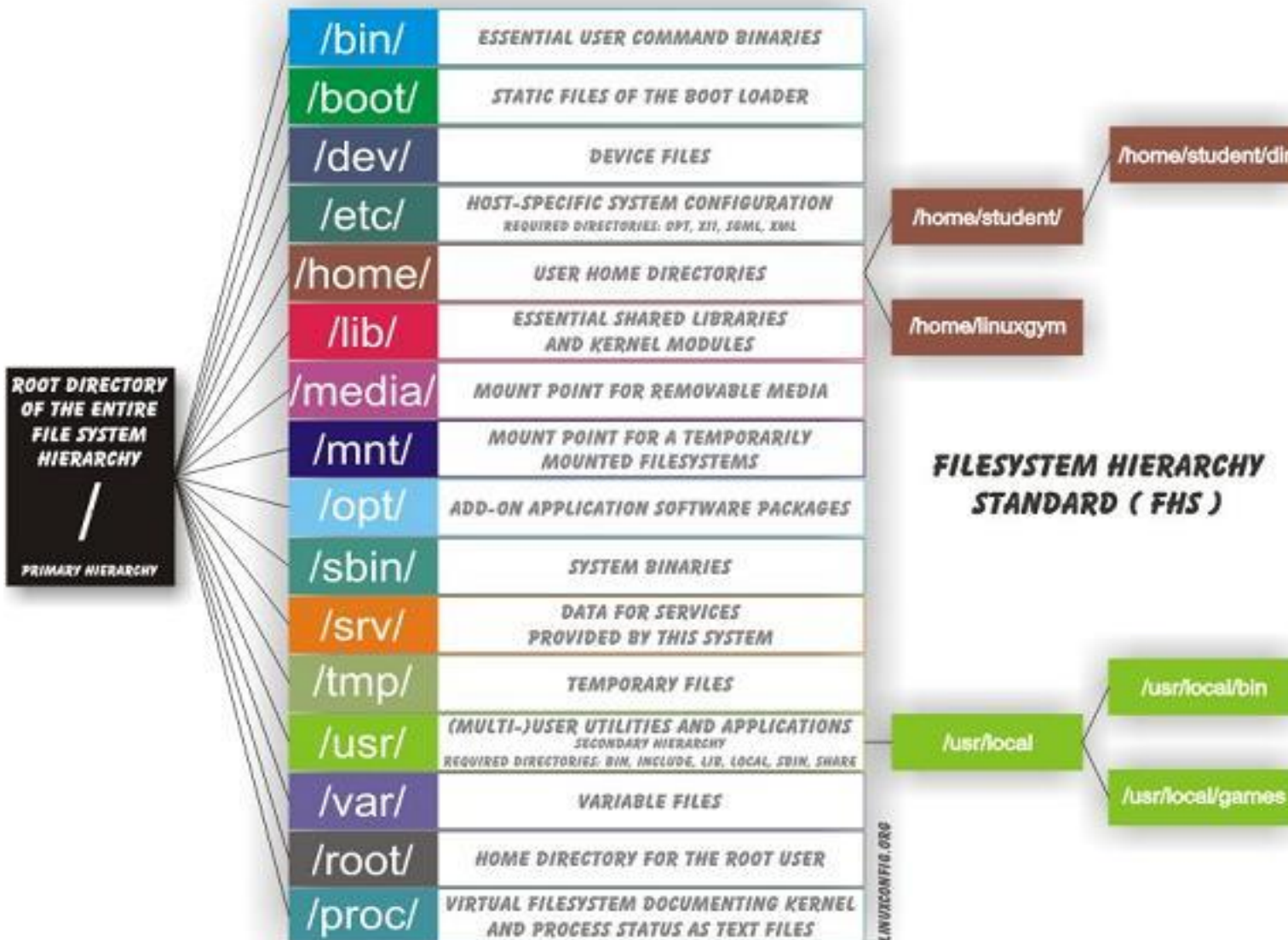
- To copy files **From** the server run the following on your workstation or laptop:

```
scp -r <your_username>@<host_name>:<path_to_directory>
```

Windows

- ▶ Use a ssh---client like Putty, Connectbox, Dropbear etc and select ssh. [More info](#) on ssh clients
- ▶ Provide the host name or IP and session name.
- ▶ Enter your identity information

Linux Directory Structure





Linux Directory Structure

- ▶ **root** is the user name or account that by default has access to all commands and files on a Linux or other Unix-like operating system. It is also referred to as the root account, root user and the superuser.
- ▶ **root directory**, which is the top level directory on a system. That is, it is the directory in which all other directories, including their subdirectories, and files reside. The root directory is designated by a forward slash (/).
- ▶ **root privileges** are the powers that the root account has on the system. The root account is the most privileged on the system and has absolute power over it (i.e., complete access to all files and commands).

Among root's powers are:

- The ability to modify the system in any way desired and to grant and revoke access permissions (i.e., the ability to read, modify and execute specific files and directories) for other users.
- To become root with su merely requires typing

```
$su or $su root
```

and supplying the root password.



Linux Directory Structure

The full names (also referred to as the **absolute pathnames**) of all of the subdirectories in the root directory begin with a forward slash, which shows their position in the filesystem hierarchy. In addition to `/bin`, some of the other standard subdirectories in the root directory include `/boot`, `/dev`, `/etc`, `/home` and `/var`.

- ▶ **/bin** is a standard subdirectory of the root directory in Unix-like operating systems that contains the executable (i.e., ready to run) programs that must be available in order to attain minimal functionality for the purposes of booting (i.e., starting) and repairing a system.
- ▶ **/etc** is the nerve center of your system, it contains all system related configuration files in here or in its sub-directories.
- ▶ **/home** Linux is a multi-user environment so each user is also assigned a specific directory that is accessible only to them and the system administrator.
- ▶ **/lib** The `/lib` directory contains kernel modules and those shared library images (the C programming code library) needed to boot the system and run the commands in the root filesystem,
- ▶ **/usr** contains all the user binaries, their documentation, libraries, header files, etc. and its supporting libraries can be found here.
- ▶ **/var** Contains variable data like system logging files, mail and printer spool directories, and transient and temporary files

Basic Commands

► ls

The ls command is used to list all the files present in the current directory.

Syntax	Example
ls [OPTIONS]	<pre>\$ ls file1 file2 file3 file4</pre>

Options	Description
-a	All. Show all the files including the hidden files.
-l	Uses a long listing format.
-h	Used along with -l. Shows the sizes in human readable format.
-s	Prints the size in blocks. 1 block = 1K
-R ...	Recursive. Lists sub directories recursively.

Basic Commands contd...

- ▶ In the Linux operating system, a hidden file is any file that begins with a ".". When a file is hidden it can not be seen with the bare `ls` command or an un-configured file manager.

Example

```
$ ls -al
total 3763888
drwxr-xr-x 63 nihithadavis nihithadavis      4096 Nov 19 19:21 .
drwxr-xr-x  5 root          root          4096 Nov 12 17:30 ..
-rw-rw-r--  1 nihithadavis nihithadavis    150 Oct 29 10:17 A
drwx-----  3 nihithadavis nihithadavis    4096 Oct  4 11:27 .adobe
-rw-r--r--  1 root          root          32421 Oct  4 16:19 arena.jpg
```

- ▶ A meta character is a character that has a special meaning and the shell never takes it at face value. * denotes any number of characters, ? denotes a single character, [] denotes a range of characters, ! denotes compliment of the following condition

Example

```
$ ls
a1 a2 carribeans fi2 gain kangaroos kiwis rain sprinboks
$ ls *in
gain rain
$ ls [aeiou]*
a1 a2
```

Basic Commands contd...

▶ **cat**

The cat command is also used to create a file. It also allows us to enter any content that we want into it.

Syntax	Example
<code>cat > <file path></code>	<pre>\$ cat > sample Hello, Welcome to the world of Linux! How are you today? Ctrl^D</pre>

▶ **mkdir**

mkdir command is used to create a directory if already not present in the linux machine.

Syntax	Example
<code>mkdir [OPTIONS] <name_of_the_directory></code>	<pre>\$ mkdir sample \$ ls sample</pre>

▶ **mv**

mv command is to rename files/directories and/or move files.

Syntax	Example
<pre><code>mv <old file name> <new file name></code> <code>mv <source file path> <destination file path></code></pre>	<pre>\$ mv sample1 firstsamp \$ mv /home/hadoop/firstsample /home/hadoop/Documents/</pre>

Basic Commands contd...

▶ **cp**

The cp command is used to copy files. cp command can also be used copy files into directories.

Syntax	Example
<code>cp <source file path> <destination file path></code>	<pre>\$ cp sample1 sample1_copy \$ cat sample1_copy This is the contents of the first file</pre>

▶ **rm**

The rm command is used to delete a file or a directory.

Syntax	Example
<p>To delete a file: <code>rm <file path ></code></p> <p>To delete a directory: <code>rm -r <path of the directory ></code></p>	<pre>hadoop@localhost sample_files]\$ ls sample1 sample2 [hadoop@localhost sample_files]\$ rm sample1 [hadoop@localhost sample_files]\$ ls sample2 [hadoop@localhost ~]\$ rm -r sample_files</pre>

The second step removes the sample1 file ,the 4th step removes the sample_files directory along with all its files

Popular Text Editors in Linux

▶ vi and vim

Non-graphical (terminal-based) editor. Vi is guaranteed to be available on any system. Vim is the improved version of vi.

▶ gedit

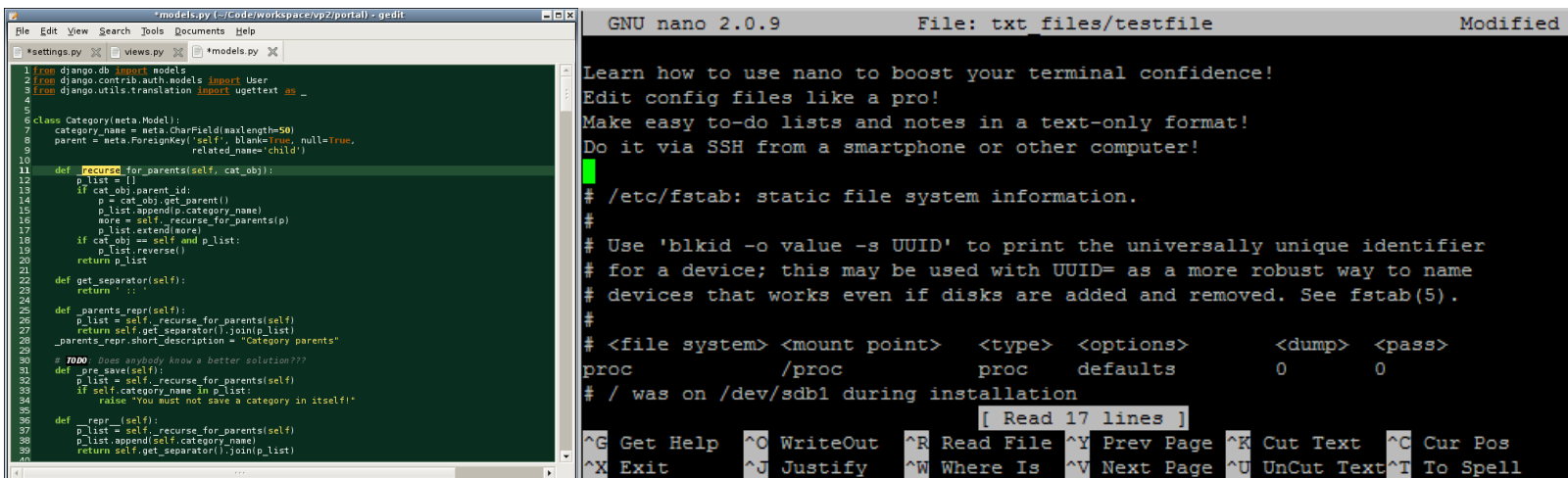
gedit is the editor supplied with the Gnome desktop environment

▶ pico

Simple terminal-based editor available on most versions of Unix. Uses keystroke commands, but they are listed in logical fashion at bottom of screen.

▶ nano

A simple terminal-based editor which is default on modern Debian systems.



```

*models.py [-Code/workspace/vp2/portal] - gedit
File Edit View Search Tools Documents Help
*settings.py views.py *models.py
1 from django.db import models
2 from django.contrib.auth.models import User
3 from django.utils.translation import ugettext as _
4
5
6 class Category(models.Model):
7     category_name = models.CharField(maxlength=50)
8     parent = models.ForeignKey('self', blank=True, null=True,
9                               related_name='child')
10
11 def recurse_for_parents(self, cat_obj):
12     p_list = []
13     if cat_obj.parent_id:
14         p = cat_obj.get_parent()
15         p_list.append(p.category_name)
16         more = self.recurse_for_parents(p)
17         p_list.extend(more)
18     if cat_obj == self and p_list:
19         p_list.reverse()
20     return p_list
21
22 def get_separator(self):
23     return ':'
24
25 def _parents_repr(self):
26     p_list = self.recurse_for_parents(self)
27     return self.get_separator().join(p_list)
28     _parents_repr.short_description = "Category parents"
29
30 # TODO: Does anybody know a better solution???
31 def _pre_save(self):
32     p_list = self.recurse_for_parents(self)
33     if self.category_name in p_list:
34         raise "You must not save a category in itself!"
35
36 def __repr__(self):
37     p_list = self.recurse_for_parents(self)
38     p_list.append(self.category_name)
39     return self.get_separator().join(p_list)
40
GNU nano 2.0.9 File: txt_files/testfile Modified
Learn how to use nano to boost your terminal confidence!
Edit config files like a pro!
Make easy to-do lists and notes in a text-only format!
Do it via SSH from a smartphone or other computer!
# /etc/fstab: static file system information.
#
# Use 'blkid -o value -s UUID' to print the universally unique identifier
# for a device; this may be used with UUID= as a more robust way to name
# devices that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dump> <pass>
proc /proc proc defaults 0 0
# / was on /dev/sdb1 during installation
[ Read 17 lines ]
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

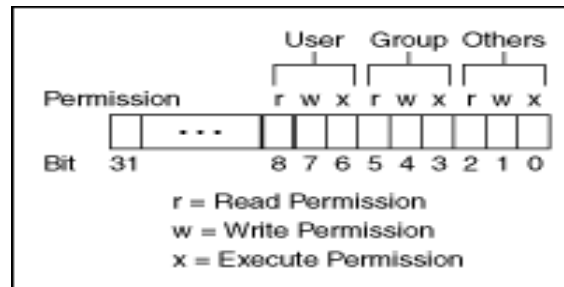
```

Setting Ownership and Permissions

► The permissions of a file signify who can access that file and for what purpose. In the `ls -l` output, the first column denotes the file permissions, a set of nine characters. There are 3 types of permissions :

read(r)	4
write(w)	2
execute(x)	1

► There are three entities to which any combinations of these permissions are assigned. These entities are the owner, the group and the rest(those outside the group).



Setting Ownership and Permissions continued...

► chmod

The chmod command is used to modify the permissions of any file. In the below example, the owner has all the permissions for the file, whereas the group and the rest have none.

Syntax	Example
<pre>chmod [numerical permissions] <file name></pre>	<pre>\$ chmod 700 zulus \$ ls -l zulus -rwx----- 1 hadoop hadoop 0 Dec 11 15:20 zulus</pre>
Syntax	Example
<pre>chmod [who] [+/-/=] [permissions] <filename></pre>	<pre>\$ chmod +x rain \$ ls -l rain -rw-rwxr-x 1 hadoop hadoop 0 Dec 11 15:20 rain</pre>

► chown

The chown command is used to change the ownership of a file. This command should be run as superuser. Here in this example the owner is changed from root to hadoop

Syntax	Example
<pre>chown [OPTIONS] [OWNER][:[GROUP]] <file name></pre>	<pre>\$ sudo chown hadoop rain \$ ls -l rain -rw-rwxr-x 1 hadoop hadoop 0 Dec 11 15:20 rain</pre>

Useful Filter Commands

▶ **sort**

The sort command is used for sorting the contents of a file. Additionally, it is also used to merge sorted files and store them in a specified file.

Syntax	Example
<pre>sort [OPTIONS] [FILE]</pre>	<pre>\$ cat names nihitha davis abc \$ sort names abc davis nihitha</pre>

▶ **cut**

The cut command is used to cut sections of a given file and return the result to standard output. The option is mandatory to use in this command.

Syntax	Example
<pre>cut OPTION [FILE]</pre>	<pre>\$ cut -c 1-4 names abc davi nihi</pre>

Useful Filter Commands continued...

▶ **wc**

The `wc` command is used to count the number of words, lines and characters in the specified file(s) . In the below example, 1 denotes the number of lines, 7 denotes the number of words and 37 denotes the number of characters.

Syntax	Example
<code>wc [OPTIONS] [FILE]</code>	<pre>\$ cat sample This the contents of the second file \$ wc sample 1 7 37 sample</pre>

▶ **grep**

The `grep` command is used for pattern matching. It searches for the given pattern in the given file and prints it. This command can be used efficiently with the help of regular expressions i.e. set of meta characters.

Syntax	Example
<code>grep [OPTIONS] PATTERN [FILE]</code>	<pre>\$ cat employee 4442 raj 25 bangalore 22000 5263 aaron 34 delhi 45000 2323 vicky 23 mumbai 21000 2830 emmanuel 39 delhi 75000 \$ grep delhi employee 5263 aaron 34 delhi 45000 2830 emmanuel 39 delhi 75000</pre>



Other filters

▶ **uniq**

Given a sorted stream of data from standard input, it removes duplicate lines of data (i.e., it makes sure that every line is unique).

▶ **tr**

Translates characters. Can be used to perform tasks such as upper/lowercase conversions or changing line termination characters from one type to another (for example, converting DOS text files into Unix style text files).

▶ **sed**

Stream editor. Can perform more sophisticated text translations than tr.

▶ **awk**

An entire programming language designed for constructing filters. Extremely powerful.

▶ **fmt**

Reads text from standard input, then outputs formatted text on standard output.

▶ **pr**

Takes text input from standard input and splits the data into pages with page breaks, headers and footers in preparation for printing.

Finding files and directories

▶ find

- The find command is used to locate files on a Unix or Linux system. find will search any set of directories you specify for files that match the supplied *search criteria*.
- You can search for files by name, owner, group, type, permissions, date, and other criteria. The search is recursive in that it will search all subdirectories too.

Syntax	Example
<code>find -name "*pattern*"</code>	<pre>\$ find -name 'sample' ./temp/sample</pre>

▶ locate

The locate command is the simplest and the quickest way to find any files or directories in the Linux operating system.

Syntax	Example
<code>locate [OPTIONS] <file(s)></code>	<pre>\$ locate mjoin /home/hadoop/mjoin.csv /home/hadoop/mjoin.csv~ /usr/local/hadoop/hadoop/logs/history/done/mjoin</pre>



Useful operations – Piping commands

The following uses three pipes to search the contents of all of the files in current directory and display the total number of lines in them that contain the string *Linux* but not the string *UNIX*:

```
$cat * | grep "Linux" | grep -v "UNIX" | wc -l
```

The following is a slightly more complex example of combining a pipe with redirection to a file:

```
$echo -e "orange \npeach \ncherry" | sort > fruit
```

The *echo* command tells the computer to send the text that follows it to standard output, and its *-e* option tells the computer to interpret each *\n* as the *newline symbol* (which is used to start a new line in the output). The pipe redirects the output from *echo -e* to the *sort* command, which arranges it alphabetically, after which it is redirected by the output redirection operator to the file *fruit*.

-Linux info

The tilde symbol (~) gets interpreted as the path to your home directory, (/) is the root directory.

File Viewing commands

▶ head

The head command is also a filter. It is used to display the first 10 lines of the file given as the argument.

Syntax	Example
<code>head</code>	<code>\$head /etc/passwd</code>

▶ tail

The tail is similar to the head command, except it outputs the last 10 lines of the file given as the argument.

Syntax	Example
<code>tail [OPTIONS] [FILE]</code>	<code>\$tail -f /etc/passwd</code>

▶ more

The more command is a filter for paging through text one screen at a time. It is useful for viewing large files.

Syntax	Example
<code>more [FILE]</code>	<code>\$more student.txt</code>

Disk related commands

▶ **df**

The df command is used to check how much of disk space is being used and how much of it lies free.

Syntax	Example
<code>df [OPTIONS]</code>	<code>\$ df</code>

▶ **du**

The du command is similar to the df command. But it does not report the disk space used and available in the whole system, but it reports the disk space used by specific files and directories.

Syntax	Example
<code>du [OPTIONS] [FILE]</code>	<code>\$ du</code>

▶ **free**

The free command is used to display the free and the used memory in the system.

Syntax	Example
<code>free [OPTIONS]</code>	<code>\$ free</code>

Process Management

▶ ps

The `ps` command is used to display all the processes that are running at any instant in the machine. Unix assigns a unique number to every process running in memory, called the PID. The output of `ps` shows the PID, the terminal from which the process was launched, the time elapsed since the launch of the process and the name of the process

Syntax	Example
<code>ps [OPTIONS]</code>	<pre>\$ ps PID TTY TIME CMD 3694 pts/0 00:00:00 bash 12125 pts/0 00:00:00 ps</pre>

▶ nice

The `nice` command is used to modify the priority of a process to execute. By default, all the processes have an equal priority of 20. This can be modified using this command. The value of this ranges from 0 to 39, a higher number would mean lesser priority and vice-versa.

Syntax	Example
<code>nice [OPTION] [COMMAND]</code>	<pre>\$ nice cat employee #The above nice command adds a value of 10 to #the priority of cat command</pre>

Process Management continued...

▶ kill

The kill command is used to terminate any running process. The PID of the process is given as the argument for this command.

Syntax

```
kill [SIGNAL_ID] PID
```

Example

```
$ ps -a u
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   COMMAND
1000         12519  0.0  0.0  18400  1672 pts/0    S+   14:52   nano start
1000         12520  0.0  0.0  22360  1268 pts/1    R+   14:52   ps -a u
$ kill 12519
$ ps -a u
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   COMMAND
1000         12525  0.0  0.0  22328  1152 pts/1    R+   14:53   ps -a u
```


Miscellaneous commands

▶ **logname**

The logname command is used to print the current user's login name.

Syntax	Example
<code>logname</code>	<pre>\$ logname nihithadavis</pre>

▶ **id**

The id command is used to print the user and group information along with the id numbers. The user and group information is displayed for the *USERNAME* or the current user if the *USERNAME* is omitted.

Syntax	Example
<code>id [OPTION] [USERNAME]</code>	<pre>\$ id uid=1000(nihithadavis) gid=1000(nihithadavis) groups=1000(nihithadavis),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),109(lpadmin),124(sambashare)</pre>

Miscellaneous commands contd...

▶ **uname**

The uname command prints the system information.

<i>Syntax</i>	Example
<code>uname [OPTIONS]</code>	<pre>\$ uname -a Linux nihitha 3.5.0-42-generic #65~precise1- Ubuntu SMP Wed Oct 2 20:57:18 UTC 2013 x86_64 x86_64 x86_64 GNU/Linux</pre>

▶ **pwd**

The pwd command is used to display the current directory that the user is working in.

pwd is the acronym for Present Working Directory.

Syntax	Example
<code>pwd</code>	<pre>\$ pwd /home/nihithadavis</pre>

Compressing with Gzip and Zip

► gzip and zip

Compressed files use less disk space and download faster than large, uncompressed files. You can compress Linux files with the open-source compression tool **Gzip** or with **Zip**, which is recognized by most operating systems.

Syntax	Example
<pre>gzip [OPTION]... [FILE]... zip [OPTION]... [FILE]...</pre>	<pre>\$gzip filename.ext #To expand a compressed file, type: \$gunzip filename.ext.gz \$zip -r filename.zip files #In this example, <i>filename</i> represents the file #you are creating and <i>files</i> represents the #files you want to put in the new file. #To extract the contents of a zip file, type: \$unzip filename.zip</pre>

Archiving with Tar

► tar

tar files place several files or the contents of a directory or directories in one file. This is a good way to create backups and archives. Usually, tar files end with the .tar extension.

Syntax	Example
<pre>tar <operation> [options]</pre>	<pre>\$tar -xvf foo.tar #verbosely extract foo.tar \$tar -xzf foo.tar.gz #extract gzipped foo.tar.gz \$tar -xzf foo.tar.gz blah.txt #extract the file blah.txt from foo.tar.gz</pre>

Installing and Uninstalling Software

▶ yum

- yum is an interactive, rpm based, package manager which can automatically perform system updates, including dependency analysis and obsolete processing based on "repository" metadata.
- It can also perform installation of new packages, removal of old packages and perform queries on the installed and/or available packages among many other commands/services. yum is similar to other high level package managers like apt-get and smart.

Syntax	Example
<code>yum [COMMAND] [OPTIONS] [PACKAGE....]</code>	<pre># yum install postgresql.x86_64 Resolving Dependencies Install 2 Package(s) Is this ok [y/N]: y Package(s) data still to download: 3.0 M...</pre>

–install package [package2].... – used to install the latest version of the package, or group of packages, while ensuring that all of the dependencies are satisfied.

–update [package1] [package2] - If run without any packages, update will update every currently installed package. If one or more packages or package globs are specified, Yum will only update the listed packages.

–remove | erase package1 [package2].. - Are used to remove the specified packages from the system as well as removing any packages which depend on the package being removed.

Installing and Uninstalling Software contd...

▶ rpm

- The rpm command is a powerful Package Manager, which can be used to build, install, query, verify, update, and erase individual software packages.
- It is similar to yum, except the rpm file has to be downloaded in the case of rpm, whereas in yum, the repo that contains the package to be installed has to be present in the /etc/yum/repos.d directory. The repo will be a link to the webpage where all the packages are present.

Syntax	Example
<pre>rpm [OPTIONS] <rpm_file></pre>	<pre>#Getting detailed information about the package httpd \$rpm -qi httpd #Determining which package installed the file /etc/httpd/conf.d \$ rpm -qf /etc/httpd/conf.d/ #Showing all the files installed my httpd \$rpm -ql httpd #Verify if the package is installed or not \$rpm -qa grep httpd</pre>

Networking commands

▶ **ifconfig**

ifconfig command is used to get information about the Network configuration or to change the configuration depending upon the options passed to it.

Executing just the command will return complete description of the all the active Network interfaces in the machine

Syntax	Example
<code>ifconfig [OPTIONS]</code>	<pre>\$ ifconfig eth0 Link encap:Ethernet HWaddr d4:be:d9:47:52:cc inet addr:172.25.1.81 Bcast:172.25.1.255 Mask:255.255.255. inet6 addr: fe80::d6be:d9ff:fe47:52cc/64 Scope:Link UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:35404 errors:0 dropped:0 overruns:0 frame:0 TX packets:19522 errors:0 dropped:0 overruns:0 carrier:0...more</pre>

Networking commands continued...

► netstat

The netstat command can be used to list the information about the current Network connections.

It is a very useful tool while dealing with any Network issues. A helpful tool for the Network administrators in tracking any suspicious activity etc.

Syntax	Example
<code>netstat</code> <code>[OPTIONS]</code>	<pre>\$ netstat Active Internet connections (w/o servers) Proto Recv-Q Send-Q Local Address Foreign Address State tcp 0 0 nihitha.local:57288 ord08s09-in-f15 ESTABLISHED tcp 0 0 nihitha.local:58250 maa03s05-in-f14 ESTABLISHED</pre>

Other Useful Commands

▶ **man**

Help on any command can be found using the man command.

Syntax	Example
<pre>man <something> # general help</pre>	<pre>\$man wc # manual on program 'word count' wc \$wc --help # short help on wc \$soap -h # for less standard programs</pre>

▶ **history**

The history command is a very useful command in Linux. It prints out by default the last 500 commands that was run on the command line.

This can be very handy in situations where one would like to revert back something that was done a few commands ago, and also to keep a track of the changes that were performed in the system.

Syntax	Example
<pre>history [OPTIONS]</pre>	<pre>\$history</pre>



Environment Variables

An environment variable is a named object that contains data used by one or more applications. In simple terms, it is a variable with a name and a value. The value of an environmental variable can for example be the location of all executable files in the filesystem, the default editor that should be used, or the system locale settings.

-To list the current environmental variables, use `printenv` to print the names and the values of each.

```
$ printenv
```

-To list specific environment variable

```
$ echo $Env_Var_Name
```

```
$ echo $PATH
```

Understanding the Path Variable: The shell uses the `PATH` variable to locate a command. `PATH` contains a list of directories separated by colons:

Example:

```
$echo $PATH
```

```
/bin:/usr/bin:/usr/local/bin
```

When you enter a command, the shell looks in each of the directories specified in `PATH` to try to find it. If it can't find the command in any of those directories, you'll see a "Command not found" message.



Writing your first script and getting it to work

To successfully write a shell script, you have to do three things:

1. Write a script-A shell script is a file that contains ASCII text. To create a shell script, you use a *text editor*. type in your first script as follows:

```
#!/bin/bash

# My first script

echo "Hello World!"
```

The first line of the script is important. This is a special clue given to the shell indicating what program is used to interpret the script. In this case, it is `/bin/bash`. Other scripting languages such as perl, awk, tcl, Tk, and python can also use this mechanism.

2. Give the shell permission to execute it-This is done with the `chmod` command as follows:

```
- $ chmod 755 my_script
```

3. Put it somewhere the shell can find it

```
- $ ./my_script
```



Executable files

The shell maintains a list of directories where executable files (programs) are kept, and just searches the directories in that list. If it does not find the program after searching each directory in the list, it will issue the famous command not found error message.

This list of directories is called your *path*. You can view the list of directories with the following command:

```
– $ echo $PATH
```

This will return a colon separated list of directories that will be searched if a specific path name is not given when a command is attempted. In our first attempt to execute your new script, we specified a pathname ("./") to the file.

You can add directories to your path with the following command, where *directory* is the name of the directory you want to add:

```
– $ export PATH=$PATH:directory
```

A better way would be to edit your `.bash_profile` file to include the above command. That way, it would be done automatically every time you log in.

linuxcommand.org/writing_shell_scripts.php for more info



Exercises

1. Create a file named "MyFirstFile" in your home folder. Open it with your favourite editor (if you don't have a favourite, pick one now). Add the following lines without quotes in the file and close it

"Lorem ipsum dolor sit amet, consectetur adipiscing elit,

sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat."

a-Find the file size of the file that you just created

b-Find the permissions, owner and group of that file

c-Change the permission to 777 and owner to root

d-Make a copy of the original file ("MyFirstFile-copy") in the current directory and move the original file to /tmp directory

e-Delete the original file



Exercises continued...

2. In the copy of the original file that you created view only the first line of the file without opening it in an editor
 - a- Replace all occurrences of 'e' with 'i' in the file without opening it in an editor
 - b- Replace all occurrences of 'i' with 'e' in the file after opening it in an editor of your choice
 - c- Create another file named "MyTextFile" from "MyFirstFile-copy" but without the first line

3. Write a shell script that will do the tasks in (2) name it as "execute-task3.sh"
 - a- Make it executable and put it in the PATH so that it can be executed when you type "execute-task3.sh" from any location in the shell.



Exercises continued...

4. Find all xml files in your system
 - a- Find all xml files in your system which has "default" in their file name
 - b- Find which processes are taking the most resources in your system
 - c- Start the sshd daemon
 - d- Find the port used by the ssh using netstat
 - e- How much free RAM is available on your system?
 - f- How much free disk space is available in your system?
 - g- Which is the largest file in your system?



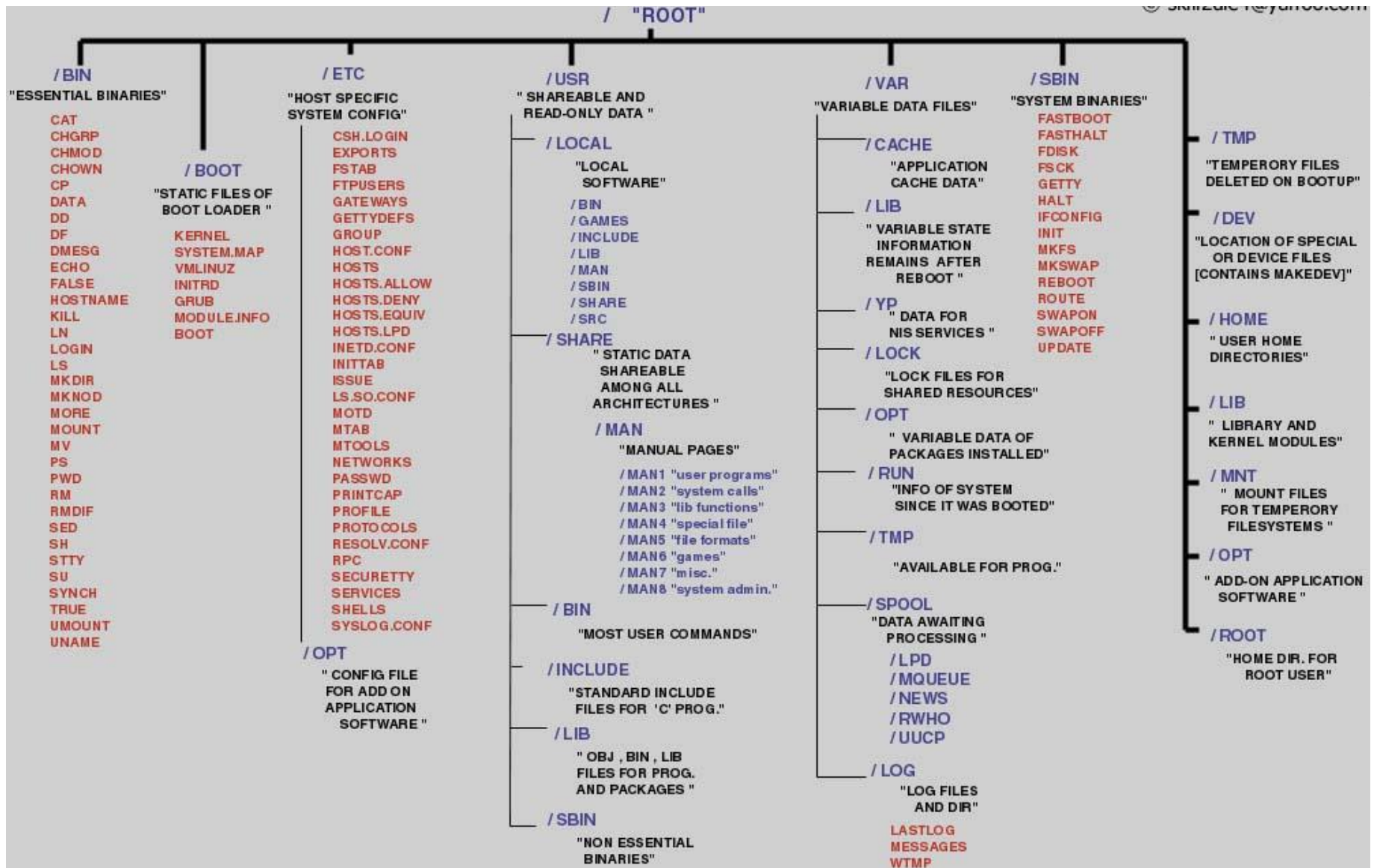
Appendix

- ▶ Linux equivalents to MS Windows and other proprietary software
- ▶ Linux Directory Structure
- ▶ Evolution of Shells in Linux
- ▶ Important Notes

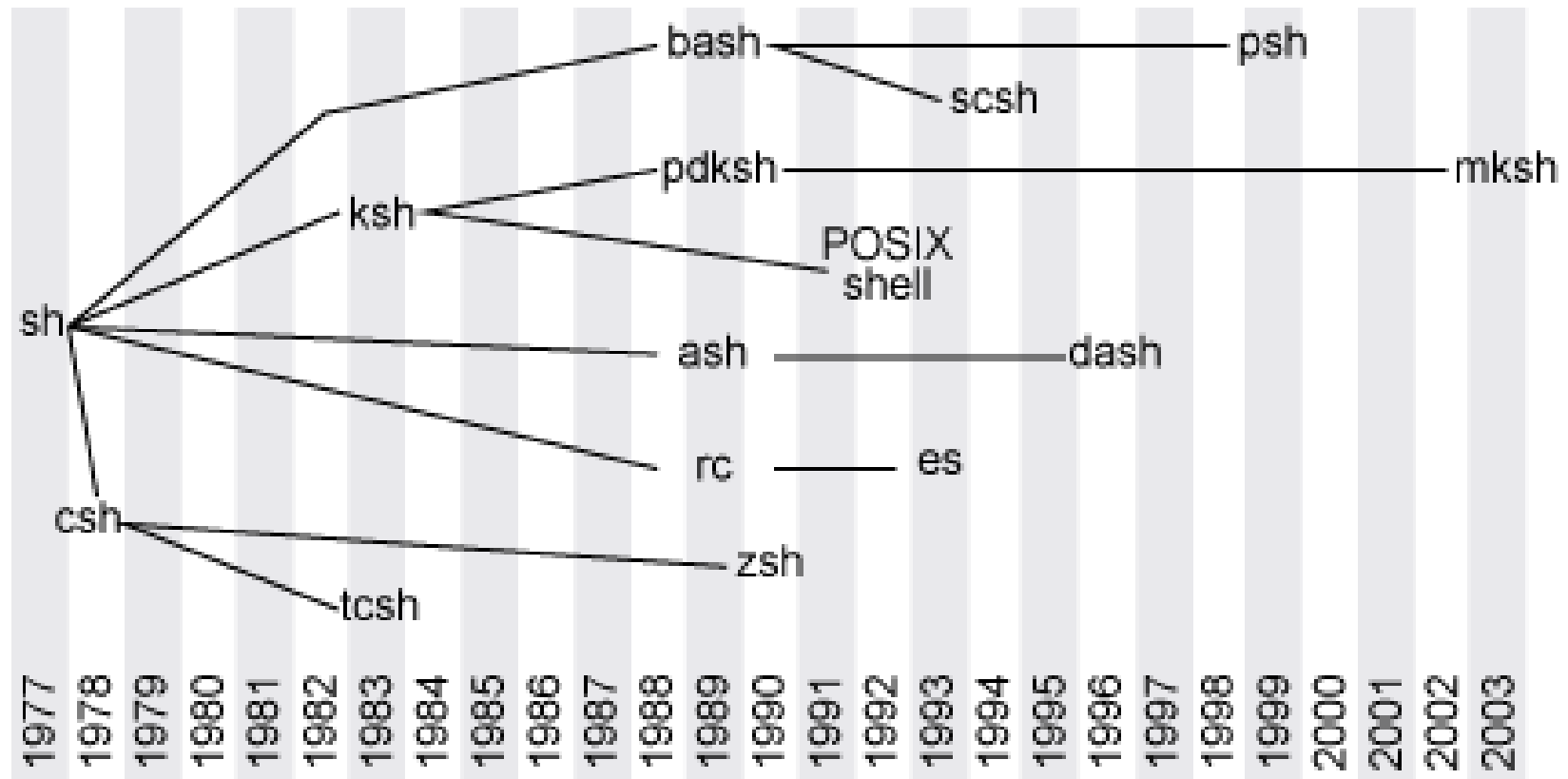
Linux equivalents to MS Windows and other proprietary software

ACDSee	Gwenview	gwenview	Photography and image viewer
Adobe Acrobat Reader	Okular	okular	Universal document viewer
Adobe Audition	Audacity	audacity	Fast, cross-platform audio editor
Adobe Lightroom	darktable	darktable	Virtual lighttable and darkroom for photographers
Adobe Photoshop	GIMP	gimp	The GNU Image Manipulation Program
iView MediaPro	Shotwell	shotwell	Digital photo organizer
Microsoft Excel	Libre Office Calc	libreoffice-calc	Office productivity suite - spreadsheet
Microsoft Office	Libre Office	libreoffice	Office productivity suite
Microsoft Outlook	Evolution	evolution	Groupware suite with mail client and organizer
Microsoft Powerpoint	Libre Office Impress	libreoffice-impress	Office productivity suite - presentation
Microsoft Project	Planner	planer	Project management application
Microsoft Visio	Dia	dia	Diagram editor
Microsoft Word	Libre Office Writer	libreoffice-writer	Office productivity suite - word processor
Notepad	Kwrite	kwrite	Simple graphical text editor
uTorrent	Ktorrent	ktorrent	BitTorrent client
Winamp	Amarok	amarok	Easy to use audio player
Windows Media Player	VLC	vlc	TOP Multimedia player and streamer

Linux Directory Structure



Evolution of Shells in Linux





Is there a way to stack commands and execute them concurrently?

If you want the commands to launch

An alternate command session and run the commands from there, simply append an ampersand

(&) symbol to the end of the first command, such as:

```
$cp /dir1/myfile.dat /dir2&; cp /dir1/myfile.dat /dev/fd0/myfile2.dat; ls -a /dev/fd0
```

In this case, Linux would launch a new session for the first command, then continue to execute the remaining commands in that session when the first one is finished.

Meanwhile,

the original session is available for your use.



Search text files for a specific character string

To search from the command line, you can use the grep utility. It is a very powerful text file search tool. You can perform wildcard searches on a specific file, but more importantly, you can search many files at once, also with wildcards. For example, say you have a directory full of text files and you know that one of them contains the phrase, “Mary had a little lamb”, but you do not know which file it is. Using the command:

```
$grep “Mary had a little lamb” *.txt
```

might result in the following output:

```
myfile.txt: Mary had a little lamb. Its fleece was white as snow. And everywhere that Mary
```



Backing up my Linux System

From a Linux standpoint, the directories you should be sure to back up include:

- /etc
- /home
- /opt
- /usr/lib
- /usr/local
- /usr/src/linux (if you have modified any of the kernel programming)
- /usr/X11R6/lib/X11 (if you are using X Windows)
- /var/spool/mail

Of course, you should also back up any directories containing important user data files. If you lack large-capacity storage, you can always back up each application's files to its own set of floppies using an archiving/compression utility, such as tar, gzip, bzip2 or zip. These and other tools can span more than one disk or tape to store all the data in one related archive. This method may not be as quick or efficient as a tape drive, but at least you will have a copy of your essential files.

It is not necessary to back up the program files themselves, as you can always just reinstall the applications from scratch, but you might want to back up the files that contain the user setup information (preferences), such as default margins and tabs for a word processor.



Changing password

- To change your own account password, Red Hat 6.2 users running KDE can click on the KDE menu and select Red Hat, then System and finally Change Password. This will bring up a panel prompting you for the current password (for the logged in user).
- From the command line, simply use the `passwd` command. It will prompt you for your current password and then the new password twice, for verification.
- To change another password, you must be logged in as root. Then tell the program which user you wish to change the password for, such as: `passwd fred`. You won't be required to know the current password; simply enter the new password twice.



What is the difference between Linux and UNIX?

- UNIX began as a proprietary operating system developed by Bell Laboratories in the 1960s. It eventually spawned a number of mutually incompatible commercial versions from such companies as Apple (Mac OS X), Digital (Digital UNIX), Hewlett-Packard (HP-UX), IBM (AIX®), NeXT (NeXTSTEP) and others.
- Linux is an attempt to create a nonproprietary operating system for the masses. It is a free, open source, UNIX-like operating system, originally begun by Linus Torvalds in 1991. “Linux” really refers to only the operating system kernel, or core. More than 200 people have contributed to the development of the Linux kernel. The rest of a Linux “distribution” package consists of various utilities, device drivers, applications, a user interface and other tools that generally can be compiled and run on other UNIX-based operating systems as well.



Bibliography

<http://gssg-www.stanford.edu>

<http://manuals.bioinformatics.ucr.edu>

<http://www.linfo.org>

<http://smallbusiness.chron.com>

<http://affexia.com>

<http://www.brighthub.com>

<http://support.suso.com>

<http://codex.wordpress.org>

<https://sudoroom.org/this-sat-713-today-i-learned-what-is-linux/>

<http://www.linuxfoundation.org/news-media/infographics/memorable-linux-milestones>

<http://www.tuxradar.com/content/best-distro-2011>

<http://www.ibm.com/developerworks/linux/library/l-linux-shells/>

<http://www.wikipedia.org>

<http://www.tecmint.com/linux-directory-structure-and-important-files-paths-explained/>



Thank You

**Chicago, IL
Bangalore, India
November, 2013
www.mu-sigma.com**

Proprietary Information

"This document and its attachments are confidential. Any unauthorized copying, disclosure or distribution of the material is strictly prohibited"