INTRODUCTION TO LINUX (PUPPY LINUX)

"Linux at the Command Line"

UNIX Introduction

- The UNIX operating system is made up of three parts;
  - the kernel, the shell and the programs.
- The kernel of UNIX is the hub of the operating system:
  - it allocates time and memory to programs and handles the file store and communications in response to system calls.
- The shell acts as an interface between the user and the kernel.

Unix

- Developed at AT&T Bell Labs
- Single monolithic kernel
  - Kernel mode
    - File system, device drivers, process management
  - User programs run in user mode
    - networking

What is Unix?

- A multi-task and multi-user Operating System
- Developed in 1969 at AT&T's Bell Labs by
  - Ken Thompson (Unix)
  - Dennis Ritchie (C)
  - Douglas McIlroy (Pipes - Do one thing, do it well)
- Some other variants: System V, Solaris, SCO Unix, SunOS, 4.4BSD, FreeBSD, NetBSD, OpenBSD, BSDI
- 64% of the world's servers run some variant of Unix or Linux. The Android phone and the Kindle run Linux.

What is Linux?

- A clone of Unix
- Developed in 1991 by Linus Torvalds, a Finnish graduate student (at the age of 21 !)
- Inspired by and replacement of Minix
- Linus' Minix became Linux
- Consist of
  - Linux Kernel
  - GNU (GNU is Not Unix) Software
  - Software Package management
  - Others

http://www.linuxdevices.com/files/misc/ibm-watchpad.jpg

What is Linux?

- Originally developed for 32-bit x86-based PC
- Ported to other architectures, eg.
  - Alpha, VAX, PowerPC, IBM S/390, MIPS, IA-64
  - PS2, TiVo, cellphones, watches, Nokia N810, NDS, routers, NAS, GPS, …

* See references at the end for the corresponding websites.

Which Linux Distribution is better?

- > 300 Linux Distributions
  - Slackware (one of the oldest, simple and stable distro.)
  - Redhat
    - RHEL (commercially support)
    - Fedora (free)
  - CentOS (free RHEL, based in England)
  - SuSe (based in German)
  - Gentoo (Source code based)
  - Debian (one of the few called GNU/Linux)
  - Ubuntu (based in South Africa)
  - Knoppix (first LiveCD distro.)
  - **PuppyLinux**

LINUX HAS MANY DISTRIBUTIONS

Run-levels in Linux Distributions

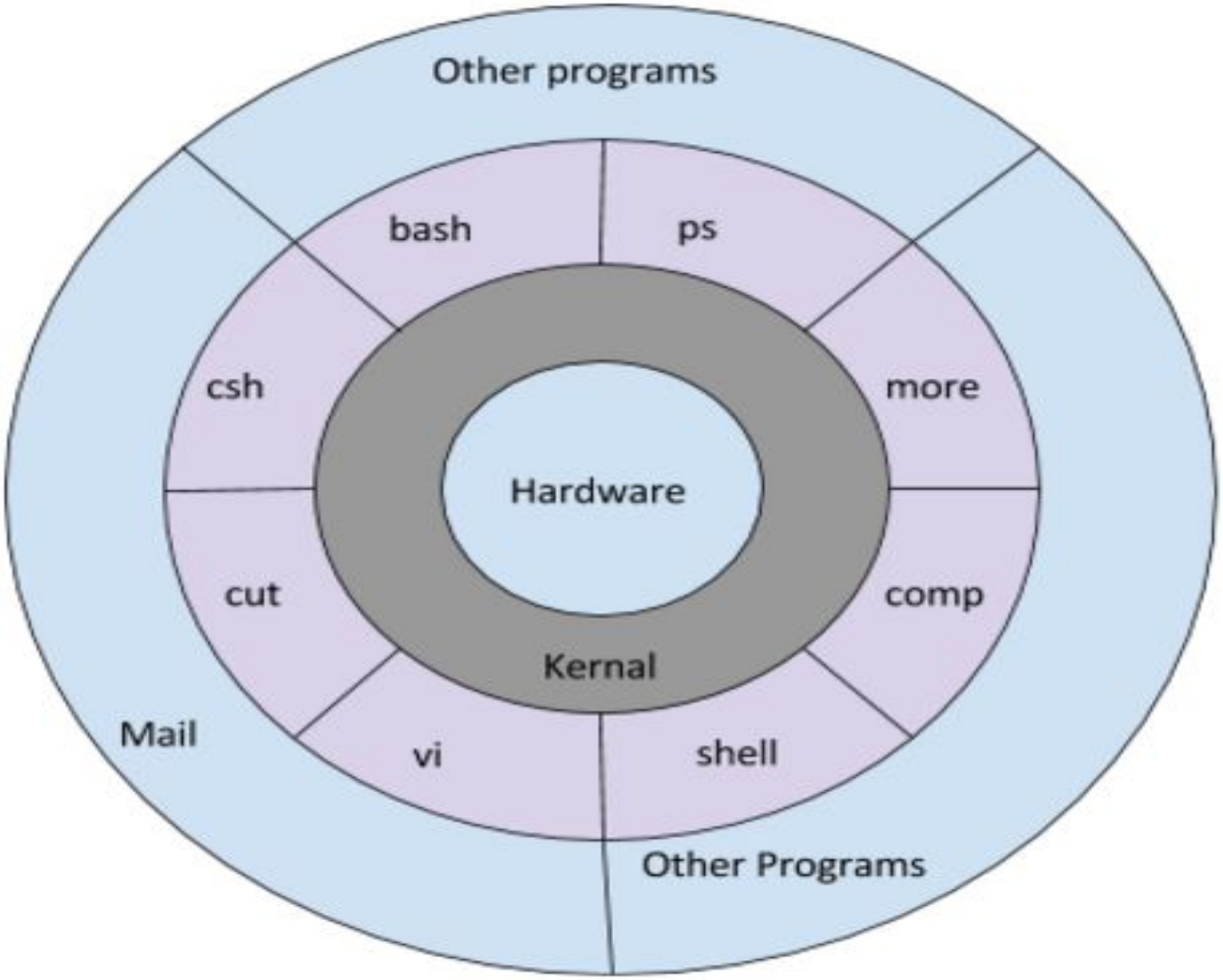
A **runlevel** is a preset operating state on a **Unix**-like operating system. ... Seven **runlevels** are supported in the standard Linux kernel (i.e., core of the operating system).

- 0 – Halt.
- 1 – Single-user text mode
- 2 – Not used (user-definable)
- 3 – Full multi-user text mode.
- 4 – Not used (user-definable)
- 5 – Full multi-user graphical mode (with an X-based login screen)
- 6 – Reboot

Reference: Linux Fundamentals, Paul Cobbaut

- Part III: Working in Command Line (root#)
  - root prompt
  - Man pages
  - Directories
  - Files
  - Linux File System

Unix Shell


Other programs
bash
ps
csh
more
Hardware
cut
comp
Kernal
Mail
vi
shell
Other Programs

Command Format

- Format: command name and 0 or more arguments:

```
% commandname [arg1] ... [argN]
```

- By % (or “root#” or $) sign means prompt here and hereafter.
- This is called as the Unix Shell
- Arguments can be
  - options (switches to the command to indicate a mode of operation) ; usually prefixed with a hyphen (-) or two (--) in GNU style
  - non-options, or operands, basically the data to work with (actual data, or a file name)

Command I/O

- Input to shell:
  - Command name and arguments typed by the user
- Input to a command:
  - Keyboard, file, or other commands
- Standard input: keyboard.
- Standard output: screen.
- These STDIN and STDOUT are often together referred to as a terminal.
- Both standard input and standard output can be redirected from/to a file or other command.
- File redirection:
  - < input
  - > output
  - >> output append

Create user – First command to learn

Open your Lab Observation Note books: Note that all commands are case sensitive and use only lower case letters:

- Command prompt: root# **adduser** <name>
  - Example: **adduser cs101 (creates a user named cs101 and**
  - Prompts for password – enter password
- To goto that user: type **login:** <enter the name>, prompts for password: <enter password> (Mypassword)
- Now you are in a secure shell and user account and your work is safe and secure.
- Type **exit** to logout of that user
- **WORK ONLY IN YOUR USER – ELSE FILES MAY BE CORRUPTED AND YOU MAY HAVE TO RE INSTALL LINUX.**

Man pages – Manual Pages (Help)

- Most Unix files and commands have pretty good **man pages** to explain their use.
- Man pages also come in handy when you are using multiple flavours of Unix or several Linux distributions since options and parameters sometimes vary.
- The first command to remember
- Contains info about almost everything :-)
  - other commands
  - system calls
  - c/library functions
  - other utils, applications, configuration files
- To read about man itself type:

```
% man man
```

- NOTE: unfortunately there's **no**

```
% man woman
```
...

The “man” command

Type these following commands in your “root#” prompt and record the answers (a short note of 3 lines) in your “root#”
man <command>

1. man whoami
2. man hostname
3. man passwd

Help command

- Command help
- Examples
  - man –help
  - cal –help
  - date –help
  - help –help

date

- Displays dates in various formats
- `% date`
- `% date -u  # in GMT`
- **% date --date=" string “**
  - **`date -date=`**"2/02/2010“
  - `date -date=“2 year ago”`
  - `date -date=“2 day ago”`
  - date --date="next tue“
  - `date -date=“2 day”`
  - `date -date=“1 year”`
  - `date -date=`

$date "+%D"
Output: 10/11/17
Command: $date "+%D %T"
Output: 10/11/17 16:13:27
Command: $date "+%Y-%m-%d"
Output: 2017-10-11
Command: $date "+%Y/%m/%d"
Output: 2017/10/11
Command: $date "+%A %B %d %T %y"
Output: Thursday October 07:54:29 17

Editor in Puppy Linux

- Text editor (Menu-document-Geany Text Editor)
  - Can be used to enter text/ programs
- Click on editor (or type geany from prompt)
- Type contents:
  - Hello World
  - I am “put your rollno and name”
  - I study First year B.Tech CSE
  - CS101 – Computer Systems Essentials
- Save file as “myfile1”
- Create 3 more files namely myfile2, myfile3, myfile4 with contents about each course of this semester (3 to 4 lines)
- We will use this later

cal

- Calendar
  - for month
  - entire year
- Years range: 1 - 9999
- No year 0
- Calendar was corrected in 1752 - removed 11 days

- `% cal` current month
- `% cal 2 2000` Feb 2000, leap year
- `% cal 2 2100` not a leap year
- `% cal 2 2400` leap year
- `% cal 9 1752` 11 days skipped
- `% cal 0` error
- `% cal 2002` whole year

clear

- Clears the screen
- There's an alias for it: Ctrl+L
- Example sequence:
  - `% cal`
  - `% clear`
  - `% cal`
  - `Ctrl+L`

sleep

- “Sleeping” is doing nothing for some time.
- Usually used for delays in shell scripts.
- `% sleep 2`
  - 2 seconds pause (you can also sleep for 2 seconds !!)

Command Grouping

- Semicolon: “;”
- Often grouping acts as if it were a single command, so an output of different commands can be redirected to a file:
- `% (date; cal; date) > out.txt`

history

- Display a history of recently used commands
- `% history`
  - all commands in the history
- `% history 10`
  - last 10
- `% history -r 10`
  - reverse order
- `% !!`
  - repeat last command
- `% !n`
  - repeat command **n** in the history
- `% !-1`
  - repeat last command = `!!`
- `% !-2`
  - repeat second last command
- `% !ca`
  - repeat last command that begins with ‘ca’

System Information Commands

date	display the current date and time
cal	display a monthly calendar
uptime	display current uptime
uname -a	display kernel information
uname -r	display kernel version
cat /proc/cpuinfo	display cpu information
dmesg	display kernel (and driver) messages
free	display memory and swap usage
df	display disk usage
du -h /PATH/DIR	display summarized disk usage for /PATH/DIR
env	displays the actual environment

File Commands

ls	directory listing
ls -al	formatted listing with hidden files and long listing
cd DIR	change directory to childdir DIR
cd /PATH/DIR	change directory to /PATH/DIR
cd ..	one dir up
cd -	change to last directory
cd	change to home
pwd	display current directory

Directory Commands

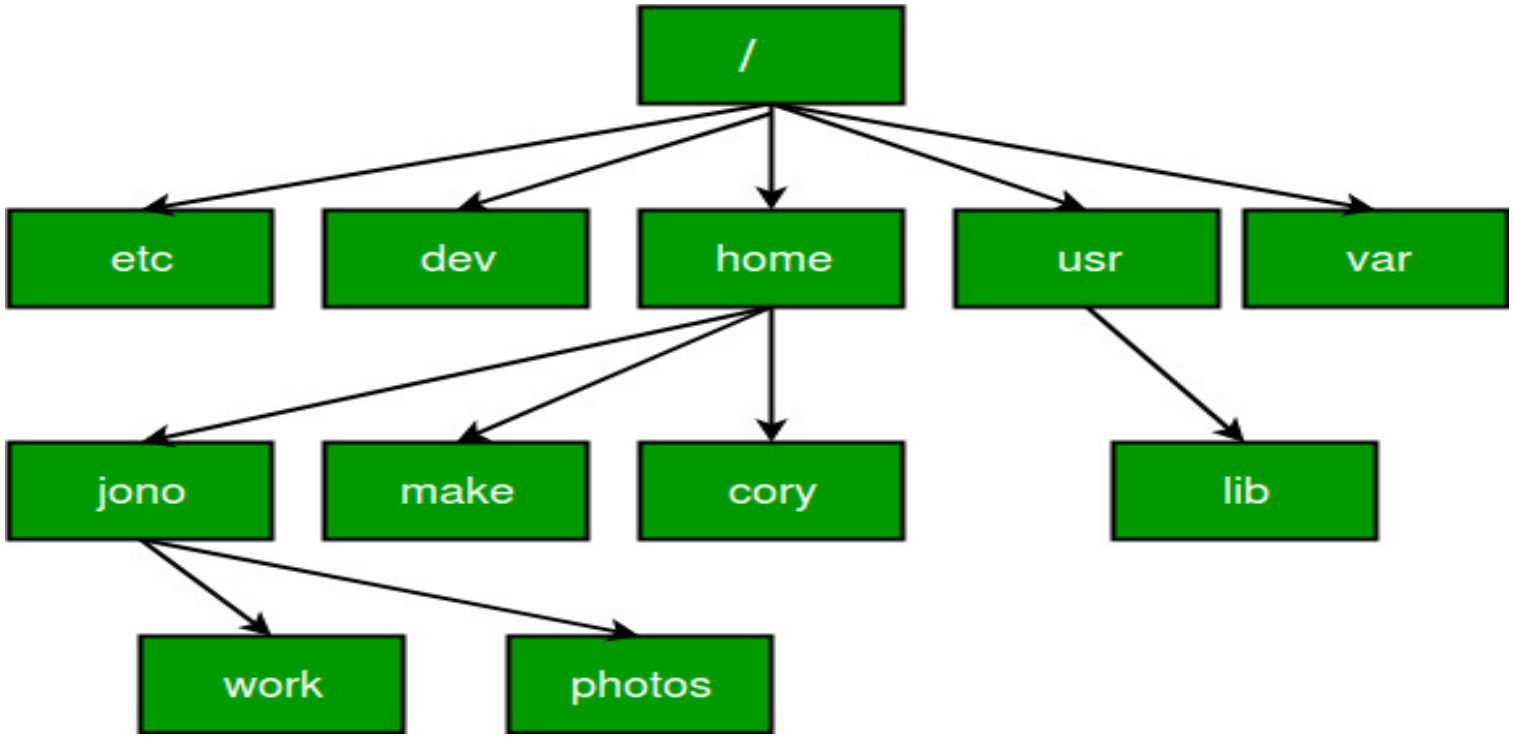
mkdir DIR	create a directory in the actual path
mkdir /PATH/DIR	create a directory in PATH
mkdir -p /PATH/DIR	create a directory in PATH even if not all dirs in PATH exist
rm FILE*	delete FILE
rm -r DIR*	delete all files in the directory and the directory named DIR

Absolute and relative paths

- A path is a unique location to a file or a folder in a file system of an OS.
- A path to a file is a combination of / and alpha-numeric characters.
- **Absolute Path-name:** An absolute path is defined as the specifying the location of a file or directory from the root directory(/).
  To write an absolute path-name:
  - Start at the root directory (/) and work down.
  - Write a slash (/) after every directory name (last one is optional)
  - Example: cat passwd – shows no such file or directory
  - Type: cd /etc , pwd, cat passwd --- shows the password file
  - Or cat /etc/passwd (absolute path of the file to be given)
  - Type cat /etc/protocols, cat /etc/pkg/pkgrc **-- Absolute path**
- The **tab key** can help you in typing a path without errors. Typing **cd /et** followed by the **tab key** will expand the command line to **cd /etc/**.

Relative path

- Relative path is defined as the path related to the present working directly(pwd).
- It starts at your current directory and **never starts with a /**
- **Sample directory structure**

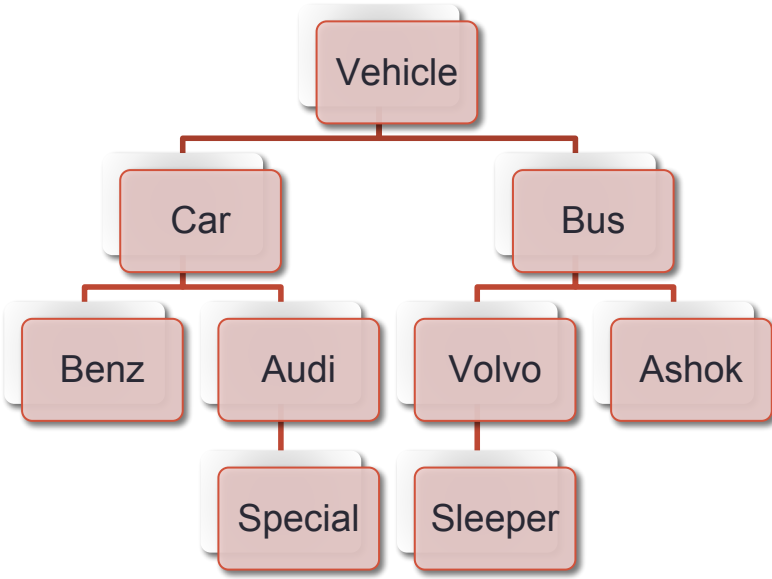


Relative Path

- UNIX offers a shortcut in the **relative pathname**– that uses either the current or parent directory as reference and specifies the path relative to it.
- A relative path-name uses one of these cryptic symbols:
  - **.(a single dot)** - this represents the current directory.
  - **..(two dots)** - this represents the parent directory.
- if we are currently in directory “/home/kt/abc” and now you can use “**..**” as an argument to “**cd**” to move to the parent directory /home/kt as : pwd (shows current working directory),
- cd .. (moves one level up)

Example – Do the following:

- Find your working directory (pwd)
- Create the following tree
- **Type mkdir vehicle, cd vehicle, pwd, mkdir car, cd car, mkdir benz….**
- Type the following Commands:
  - pwd, goto Vehicle, pwd, goto Car, pwd, goto Benz, pwd
- How to goto Audi?
- How to goto Car?
- How to goto Sleeper?
- How to goto Special?
- How to return to your home?
- Write both absolute and relative path



File Commands

cp FILE1 FILE2	copy FILE1 to FILE2
mv FILE1 FILE2	rename or move FILE1 to FILE2 if FILE2 is an existing directory, moves FILE1 into directory FILE2
touch FILE	create FILE
command > FILE	(re-)places output of command in FILE
command >> FILE	appends output of command to FILE
command < FILE	use FILE as input for command
cat FILE	display the content of FILE
tac FILE	display the content of FILE from last to first line
head FILE or head –n FILE	display the first 10 lines of FILE or n lines from FILE
tail FILE	display the last 10 lines of FILE
tail -f FILE	display the content of FILE as it grows, starting with the last 10 lines
file FILE	gives info about FILE -- nature of file

To learn more commands -- Create files

- Create myfile with the following contents:

echo “I am fine, I do CSE” > myfile

echo “I am good, I like CSE” >> myfile

echo “I like drawing” >>myfile

echo “I like reading” >>myfile

echo “I like writing” >> myfile

- Show the contents of myfile: cat myfile

To learn more commands

- cat and tac -- concatenate (display) – **tac** (last line to first line)
  - Ex: cat myfile – display file contents, tac myfile – display last to first
- **grep** is to filter lines of text containing (or not containing) a certain string.
  - Ex: grep “CSE” myfile – displays all lines that contain CSE from myfile
  - Ex: grep “like” myfile – displays all lines that contain “like” from myfile
- **Word Count – wc** options <filename>
  - Gives the no. of lines, words and characters in the file
  - Ex: wc myfile
  - Ex: wc –l myfile -- no. of lines only
  - Ex: wc –w myfile – no. of word
  - Ex: wc –c myfile – no. of characters

Some more ….

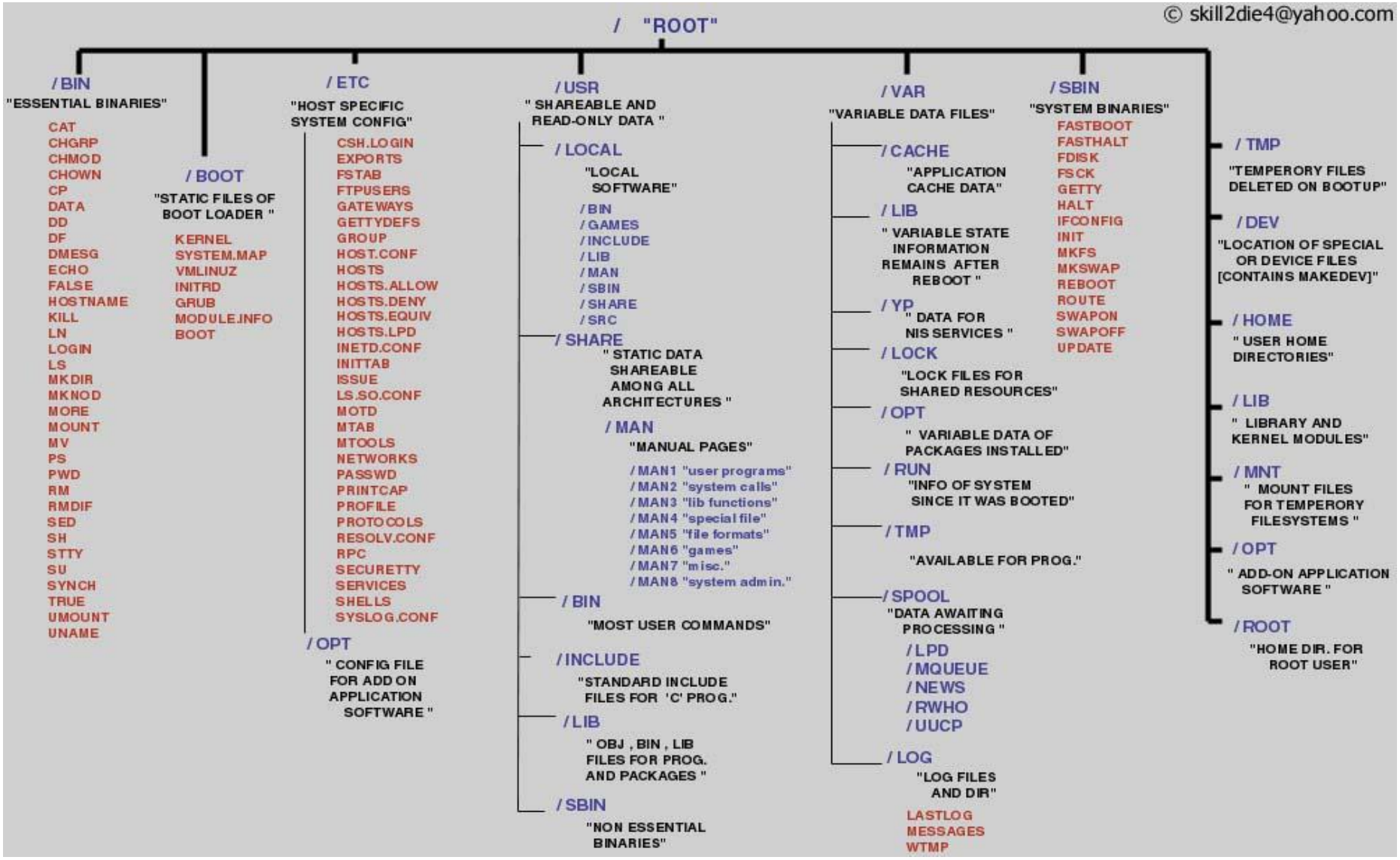
- Sorting: **sort** filter will default to an alphabetical sort.
  - Ex: sort myfile – sorts based on the alphabetical order
  - Ex: cat /etc/passwd
  - Ex: sort /etc/passwd
- **Unique lines -- uniq** you can remove duplicates from a **sorted list**.
  - Ex: uniq myfile – displays only unique lines
- Difference – **diff** command used to display the differences in the files by comparing the files line by line. it tells us which lines in one file have is to be changed to make the two files identical. (a – add, c – change, d – delete)
- Ex: head -2 myfile > myfile1 (first two lines of myfile goes to myfile1)
- **diff myfile myfile1 -- shows difference between myfile and myfile1 in terms of where to add, change and delete**

Practice the following:

- tac myfile > myfile1
- echo “periodical test 1” >> myfile1
- echo “cs101 – csessentials” >> myfile1
- ls –l
- ls –lh
- wc myfile
- sort myfile
- grep “cse” myfile
- uniq myfile myfile1
- diff myfile myfile1
- head -3 myfile
- tail -4 myfile1
- head -5 myfile >myfile2
- tail -3 myfile >myfile3
- mkdir d1
- mkdir d2
- mkdir d3
- cd d1
- pwd

- cal (all options)
- date (all options)
- cd (all options, absolute and relative path)
- sleep 2
- sort myfile
- cal:date:echo :myname” > f1
- cp f1 f2
- cp f2 f3
- rm f3
- mv f1 f3

The Linux File System



Permissions in Linux

Permission Groups

Each file and directory has three user based permission groups:

- **owner** - The Owner permissions apply only the owner of the file or directory, they will not impact the actions of other users.
- **group** - The Group permissions apply only to the group that has been assigned to the file or directory, they will not effect the actions of other users.
- **all users** - The All Users permissions apply to all other users on the system, this is the permission group that you want to watch the most.

Permission Types

Each file or directory has three basic permission types:

- **read** - The Read permission refers to a user's capability to read the contents of the file.
- **write** - The Write permissions refer to a user's capability to write or modify a file or directory.
- **execute** - The Execute permission affects a user's capability to execute a file or view the contents of a directory.

Viewing the Permissions

- You can view the permissions by the output of the ***"ls -l"*** command
- displayed as: ***_rwxrwxrwx 1 owner:group***
- User rights/Permissions
  - The first character that I marked with an underscore is the special permission flag that can vary. (d- directory, l-linked file…)
  - The following set of three characters (rwx) is for the owner permissions.
  - The second set of three characters (rwx) is for the Group permissions.
  - The third set of three characters (rwx) is for the All Users permissions.
- Following that grouping since the integer/number displays the number of hard links to the file.
- The last piece is the Owner and Group assignment formatted as Owner:Group.

Modifying the Permissions of a file: chmod

- The Permission Groups used are:
  - **u** - Owner
  - **g** - Group
  - **o** - Others
  - **a** - All users
- The potential Assignment Operators are + (plus) and - (minus); these are used to tell the system whether to add or remove the specific permissions.
- The Permission Types that are used are:
  - **r** - Read
  - **w** - Write
  - **x** - Execute

Examples:

- Create a file named as: file1
- Type ls -l, you get this:
  - **_rw_rw_r_ _** which means that the owner, group and all users have only read permission.
  - Now we want to remove the read and write permissions from the all users group.
- To make this modification you invoke the command: ***chmod a-r file1***
- To add the permissions above you would invoke the command: ***chmod a+rw file1***
- As you can see, if you want to grant those permissions you would change the minus character to a plus to add those permissions.

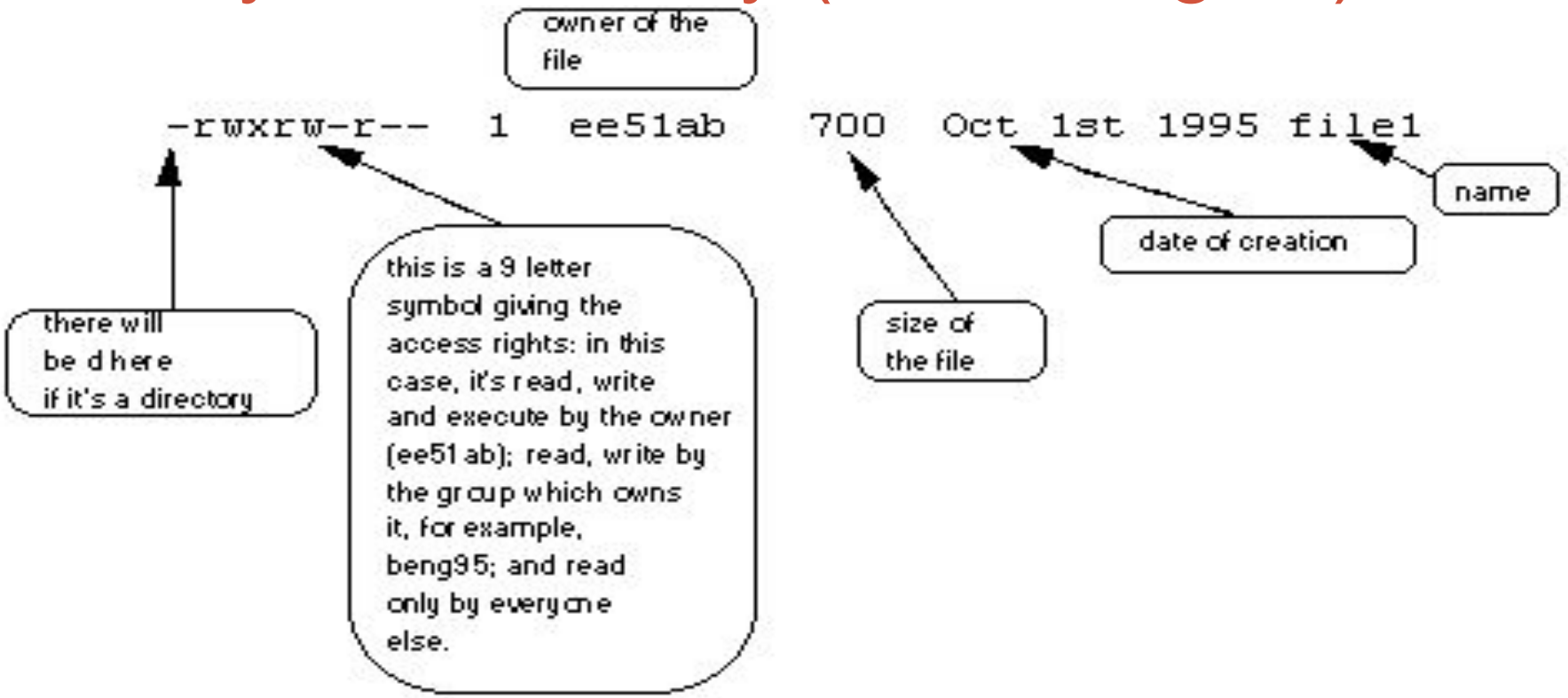
Using Binary References to Set permissions

- The numbers are a binary representation of the rwx string.
  - ***r*** = *4*
  - ***w*** = *2*
  - ***x*** = *1*
- **Example: chmod 640 file1**, which means that the owner has read and write permissions (4+2), the group has read permissions (4), and all other user have no rights (0) to the file.
- **What is chmod 777 file1?**
- **What is chmod 600 file1?**
- **What is chmod 000 file1?**

Owners and Groups

- Change owner and Groups: -- can be done only by super user/ root/ admin
  - chown “username” filename(s)
  - chgrp “groupname” filename(s)

File system security (access rights)



-rwxrwxrwx	a file that everyone can read, write and execute (and delete).
-rw-------	a file that only the owner can read and write - no-one else can read or write and no-one has execution rights (e.g. your mailbox file).

Unix File Permissions

- File type, owner, group, others

```
drwx------   2 jjoshi  isfac  512  Aug 20  2003 risk
 management
lrwxrwxrwx   1 jjoshi  isfac   15  Apr  7 09:11 risk_m->risk
 management
-rw-r--r--   1 jjoshi  isfac 1754  Mar  8 18:11 words05.ps
-r-sr-xr-x   1 root    bin   9176  Apr  6  2002 /usr/bin/rs
-r-sr-sr-x   1 root    sys   2196  Apr  6  2002
 /usr/bin/passwd
```

- File type: regular -, directory d, symlink l, device b/c, socket s, fifo f/p
- Permission: r, w, x, s or S (set.id), t (sticky)
- While accessing files
  - Process EUID compared against the file UID
  - GIDs are compared; then Others are tested

Exercise

- Work with Chapter 8.8
- Write your results in Lab Observation Notebook
- Verify your results with Chapter 8.9