RHSA1

Red Hat System Administration I

Information Technology Institute

COURSE MATERIALS

You can access the course materials via this link

https://goo.gl/ezCT7j

DAY 1 CONTENTS

- Free/Open Source Software and Licenses.
- Linux History.
- Linux Components.
- Installation
- Basic Commands
- Linux Documentation
- File and Directory Basics

WHAT IS FOSS?

- Free/Open Source Software (FOSS) provides many freedoms, including the ability to:
  - View the source code used to compile programs
  - Make modifications
  - Distribute these modifications.
- Most FOSS is covered under a public license. The most common public license is the GNU General Public License (GPL).

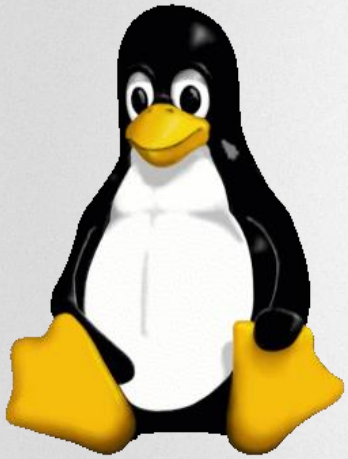
FOSS LICENSES

- An open-source license is a type of license for computer software and other products that allows the source code, blueprint or design to be used, modified and/or shared under defined terms and conditions.
- Examples:
  - GPL, LGPL, Apache, Mozilla Public License and BSD.

LINUX HISTORY

- Unix first version created in Bell Labs in 1969
- Unix flavors
  - IBM->AIX
  - Hewlett-Packard->HP/UX
  - Sun-> Solaris
  - Silicon Graphics->IRIX
- Operate in a same manner
- Offer the same standard utilities and commands
- Linus Torvalds
- Finished his college in 1991
- Created Linux kernel

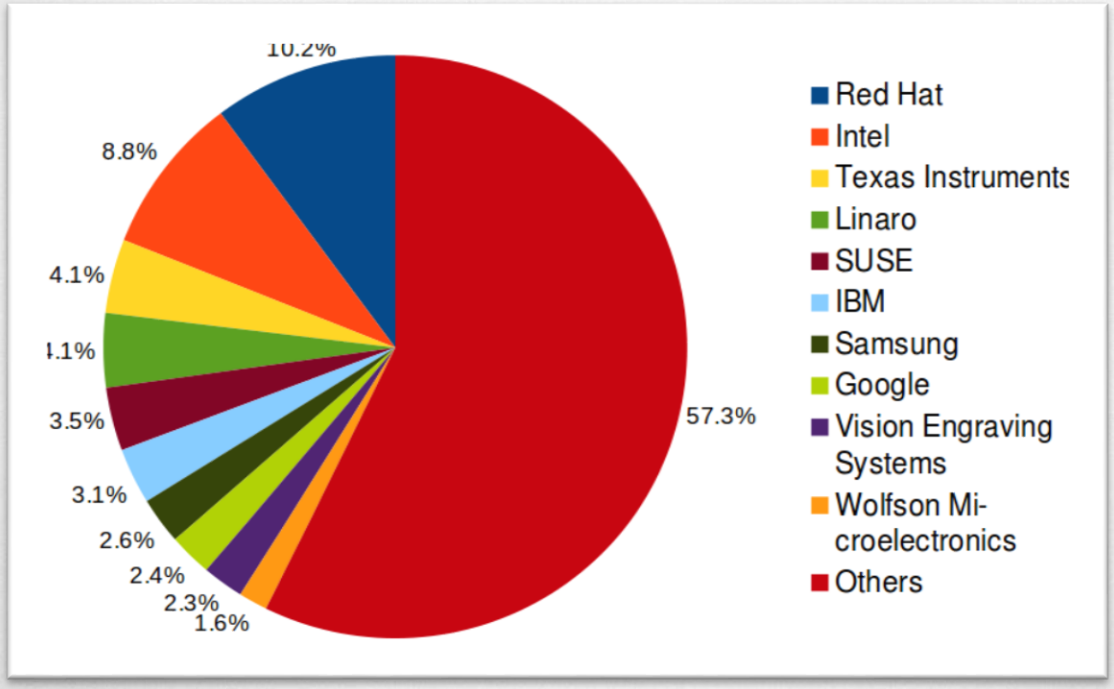
DISTROS



Linux Flavors

http://distrowatch.com

CONTRIBUTORS



WHY LINUX?

- Linux is growing in the home users sector and the dominant of the professional and servers sector.
- Internet service providers (ISPs), e-commerce sites, and other commercial applications all use Linux today and continue to increase their commitment to Linux.

WHY RED HAT?

- More than 90% of Fortune Global 500 companies use Red Hat products and solutions*.
- The most demanding applications run better on Red Hat Enterprise Linux.
- RHEL scales well, and is more reliable.
- RHEL is secure.
- Red Hat partnership with hardware vendors.
- Red Hat training and support.

* http://money.cnn.com/magazines/fortune/global500/2013/full_list/

TYPES OF INSTALLATION

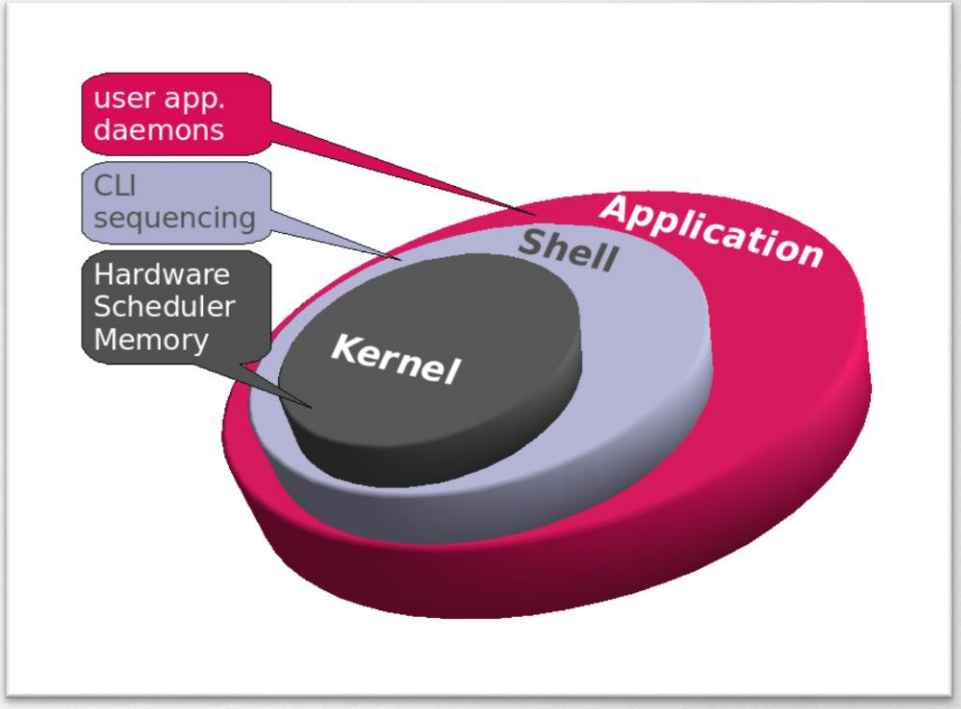
- Kickstart Mode
  - Permits automated installation
- Graphical Installation
- Text Based Installation

LINUX COMPONENTS

- **Kernel**
  - Is the core of the operating system.
  - Contains components like device drivers.
  - It loads into RAM when the machine boots and stays resident in RAM until the machine powers off.
- **Shell**
  - Provides an interface by which the user can communicate with the kernel.
  - “bash” is the most commonly used shell on Linux.
  - The shell parses commands entered by the user and translates them into logical segments to be executed by the kernel or other utilities.

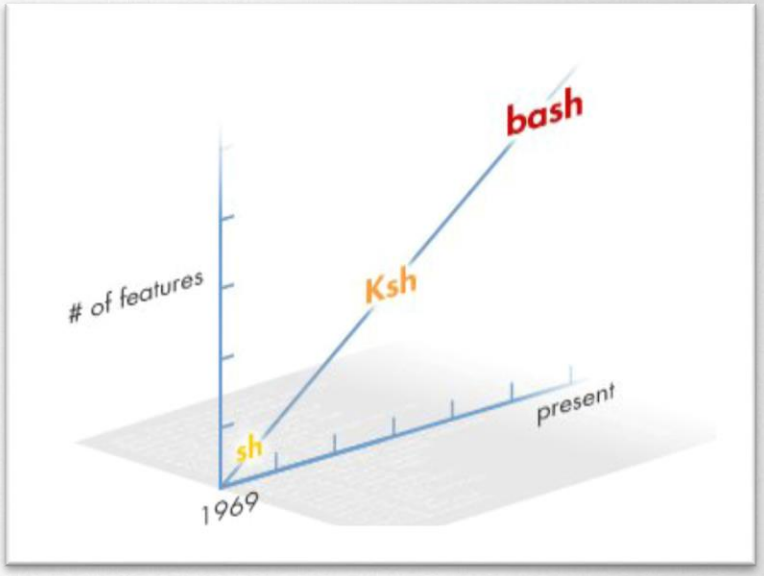
LINUX COMPONENTS

- **Terminal**
  - Gives the shell a place to accept typed commands and to display their results



COMMAND-LINE SHELLS

- There are lot of shells as :
  - Bourn Shell (sh),
  - Korn Shell (ksh),
  - C Shell (csh) and
  - Bourn Again Shell (bash).



- They have different features that will be discussed later.

RUNNING COMMANDS

- Commands have the following syntax:

```
command [options] [arguments]
```

- Each item is separated by a space.
- Options modify the command's behavior.
- Arguments are files name or other information needed by the command.
- Separate commands with semicolon (;).

EXAMPLES

uname

```
Linux
```

uname -n

```
host1
```

uname -a

```
Linux host1 ……
```

EXAMPLES

cal

```
September 2010
 S  M Tu  W Th  F  S
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

EXAMPLES

cal 5 2004

```
May 2004
  S   M Tu   W Th   F   S
                      1
  2   3   4   5   6   7   8
  9  10  11  12  13  14  15
 16  17  18  19  20  21  22
 23  24  25  26  27  28  29
 30  31
```

cal; uname

cal 5 2002; date; uname

INTERRUPTING EXECUTION

- To interrupt a command that's taking too long to execute, use **[Ctrl]-c**.
- Occasionally, you might enter a command without an argument that expects input to come from the keyboard. In this case, use **[Ctrl]-d** to terminate the command.

LINUX DOCUMENTATION

Manual page consists of:

- Name
  - The name of the command and a one-line description
- Synopsis
  - The syntax of the command
- Description
  - Explanation how the command works and what it does
- Files
  - The file used by the command
- Bugs
  - Known bugs and errors
- See also
  - Other commands related to this one

LINUX DOCUMENTATION

`man -k keyword`

Shows the commands that have manual pages that contains any of the given keywords.

`man -s keyword`

`whatis command`

Shows the commands one line description

LINUX DOCUMENTATION

`--help` Option

- Another way to get help about a command.
- help is built in the command itself (if supported).

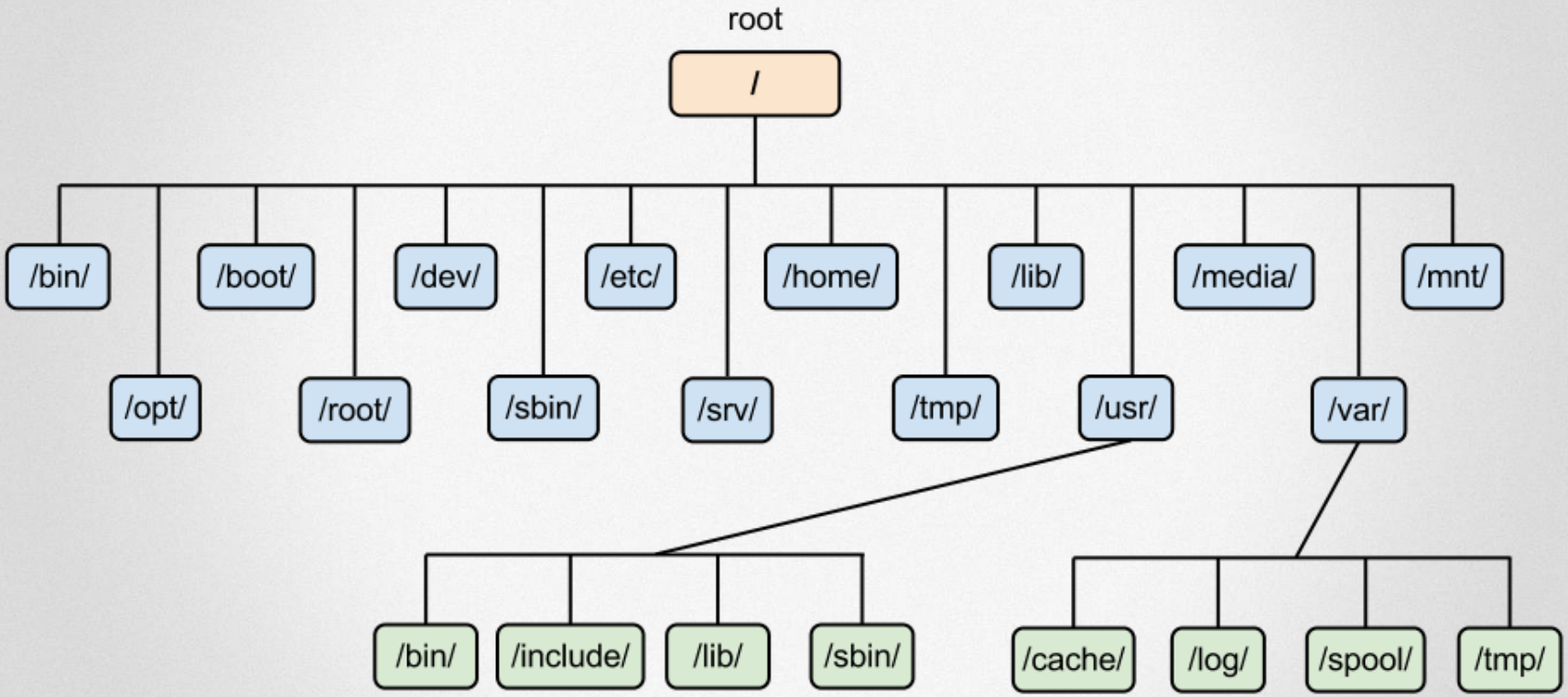
DIRECTORIES

- Think of
  - File system as a building
  - Directory is a room
  - File is a desk
- The current working directory is the room you are.
- To find out where you are at any time

```
pwd
```

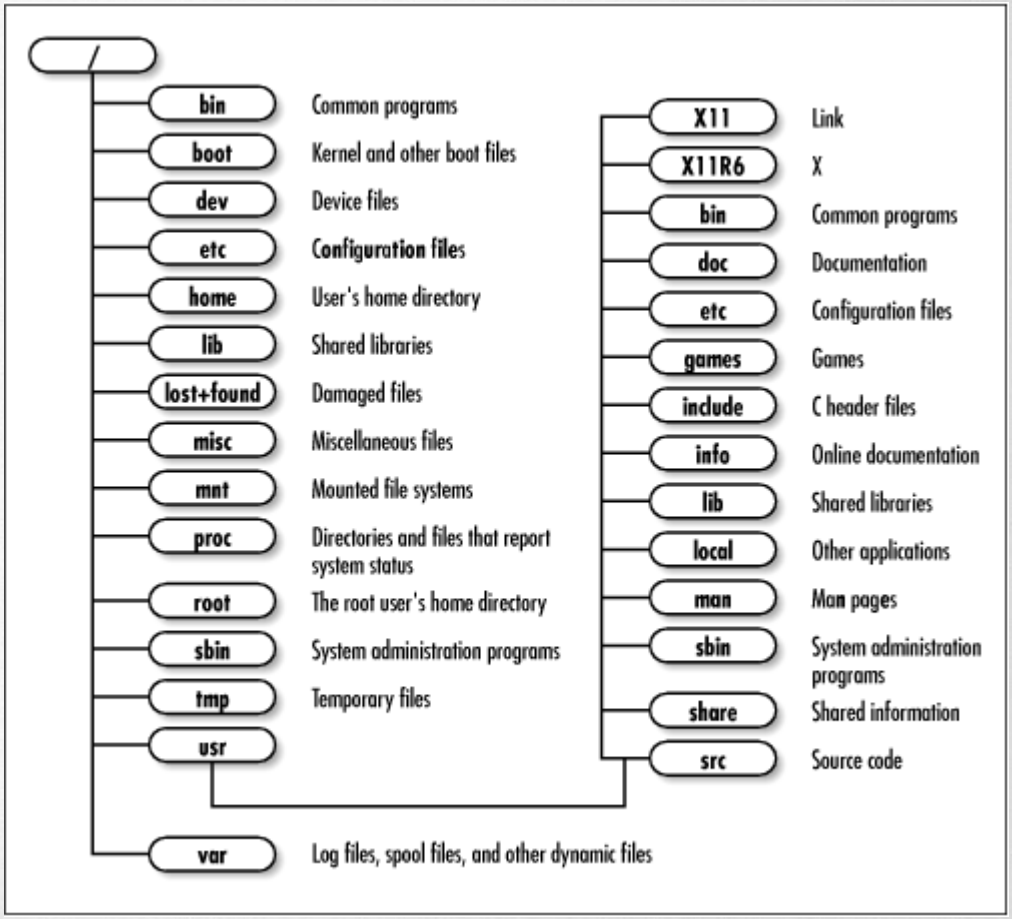
```
/home/guest
```

DIRECTORIES TREE



DIRECTORIES

- Pathnames
  - Absolute pathname
  - Relative pathname



CHANGING DIRECTORIES

- To move from directory to directory on the system

```
cd /home/user1/work
cd ..
cd ~
cd -
```

LISTING DIRECTORY CONTENTS

```
ls
```

```
dir1   dir2    file1
dir3   file2      file3
```

```
ls /home/user1/dir1
```

```
f1    f2
```

```
pwd
```

```
/home/user1
```

```
ls dir1
```

```
f1    f2
```

LISTING DIRECTORY CONTENTS

ls -a dir1

```
.      .f1   f1
..     .f2   f2
```

ls -l dir1

```
total 2
-rw-r--r-- 1 islam islam 20 2 May 21 16:11 f1
-rw-r--r-- 1 islam islam 20 0 May 21 16:11 f2
```

ls -F

```
dir1/    dir2/    file1
dir3/    file2*       file3@
```

LISTING DIRECTORY CONTENTS

ls -ld dir1

```
drwxr-xr-x 2 islam islam 20 512 May 21 16:06 dir1
```

ls -R

```
.:
dir1    dir2      file1
dir3    file2     file3
./dir1:
f1    f2
./dir2:
./dir3:
```

FILE NAMING

- File names may be up to 255 characters.
- There are no extensions in Linux
- Avoid special characters as >< ? * # '
- File names are case sensitive

VIEWING FILE CONTENT

`cat fname`

`more fname`

- Scrolling keys for the more command
  - Spacebar: moves forward on screen
  - Return: scroll one line at a time
  - b: move back one screen
  - /string: search forward for pattern
  - n: find the next occurrence
  - q: quit and return to the shell prompt

`head –n fname`

`tail [-n|+n] fname`

FILE GLOBING

- When typing commands, it is often necessary to issue the same command on more than one file at a time.
- The use of wildcards, or “metacharacters”, allows one pattern to expand to multiple filenames

METACHARACTERS

- Asterisk(*): represents 0 or more character, except leading (.)

Example:

```
ls f*
file.1 file.2 file.3 file4
file1 file2 file3 fruit

ls *3
file.3 file3
dir3:
moon planets space sun
```

METACHARACTERS

- Question mark(?) character represents any single character except the leading (.)

Examples

```
ls file?
```

```
file4 file1 file2
```

```
ls z?
```

```
z?: No such file or directory
```

- Square bracket([]): represent a range of characters for a single character position.

Example

```
ls [a-f]*
```

```
ls [pf]*
```

METACHARACTERS

```
ls -a
. .. .profile abm bam bat battle project
ls -l b*
-rw-r----- 1 sgs 16 Feb 12 11:04 bam
-rw-r----- 1 sgs 12 Feb 12 11:05 bat
-rw-r----- 1 sgs 19 Feb 12 11:06 battle
ls *
abm bam bat battle project
ls  .*
. .. .profile
ls *m
abm bam
```


METACHARACTERS

```
ls  ???
```

```
abm bam bat
```

```
ls  ?a?
```

```
bam bat
```

```
ls  ?a*
```

```
bam bat battle
```

```
ls *a*
```

```
abm bam bat battle
```


METACHARACTERS

```
ls [ab]*
abm bam bat battle
ls -l [ab]m
ls: "[ab]m: No such file or directory
ls  [a-zA-Z]*
abm bam bat battle project
```


FILE & DIR. MANIPULATION

- Coping Files and Directories

```
cp options source(s) target
```

Option	Description
-i	Prevents you from accidentally overwriting existing files or directories
-r	Copy a directory including the contents of all subdirectories

FILE & DIR. MANIPULATION

- Moving and Renaming Files and Directories

```
mv options source(s) target
```

Option	Description
-i	Prevents you from accidentally overwriting existing files or directories

FILE & DIR. MANIPULATION

- To create files

```
touch file(s)_name
```

- To create directories

```
mkdir [-p] dir(s)_name
```


FILE & DIR. MANIPULATION

- To remove files

```
rm [-i] file(s)_name
```

- To remove directories

```
rmdir dir(s)_name
```

```
rm [-r] dir(s)_name
```