Ubuntu Fundamentals

Freedom is a choice

Course Objectives

- Gain sufficient skills to perform Ubuntu system administration tasks.

Course Prerequisites

- You don't need any experience with Linux to take this course
- You should have some familiarity with computers

Agenda

- Open Source philosophy
- History
- Why Ubuntu?
- Getting Started
- How to fish?
- Files and Directories

Open Source Philosophy

Open Source Philosophy

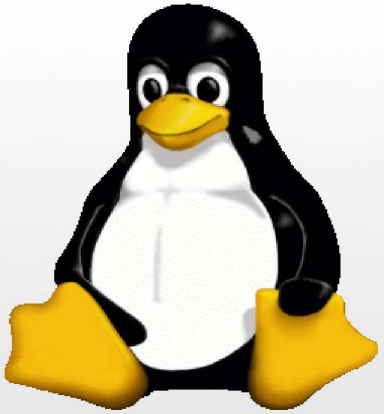
- Open Source Software (OSS) provides many freedoms, including the ability to:
  - View the source code used to compile programs
  - Make modifications
  - Distribute these modifications
- Where is the benefit ?
  - Customers are usually willing to pay for training, support and consultation

History

History

History

- 1991: The Linux kernel is publicly announced on 25 August by Linus Torvalds.
- 1992: The Linux kernel is re-licensed under the GNU GPL.
- 1993: Over 100 developers work on the Linux kernel.
- 1998: Many major companies such as IBM, Compaq and Oracle announce their support for Linux.
- 10-2017: Version 4.13 of the Linux kernel is released.



History

Linux Distribution
http://distrowatch.com

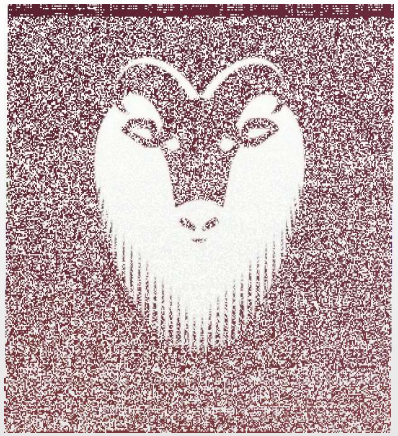
- 2016: Google's Linux-based Android claims 75% of the smart phone market share.

History

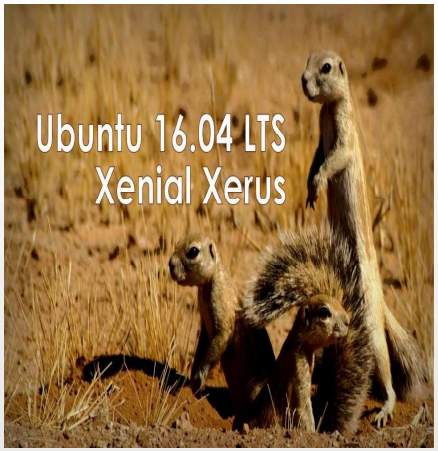
- Ubuntu based on Debian GNU/Linux distribution and distributed as free and open source software by Canonical Ltd UK.
- It is named after the Southern African philosophy of Ubuntu ("humanity towards others").
- Ubuntu is designed primarily for desktop usage, Web statistics suggest that Ubuntu's share of Linux desktop usage is about 50 percent, and upward trending usage as a web server.
- 2016: Ubuntu claims 30,000,000 users.

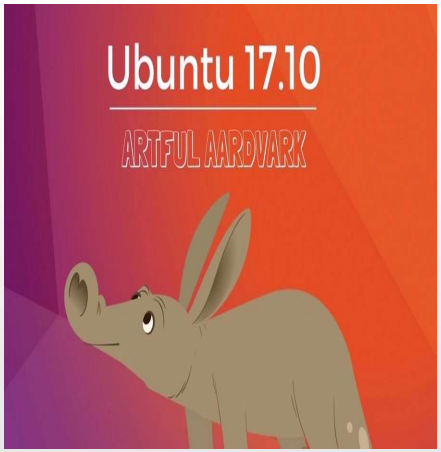
Ubuntu Releases

- The Ubuntu team broke new ground in committing to a program of scheduled releases on a predictable six-month basis. It was decided that every fourth release, issued on a two-year basis, would receive long-term support (LTS).
- LTS releases are typically used for large-scale deployments.



Ubuntu 14.04 LTS (Trusty Tahr)



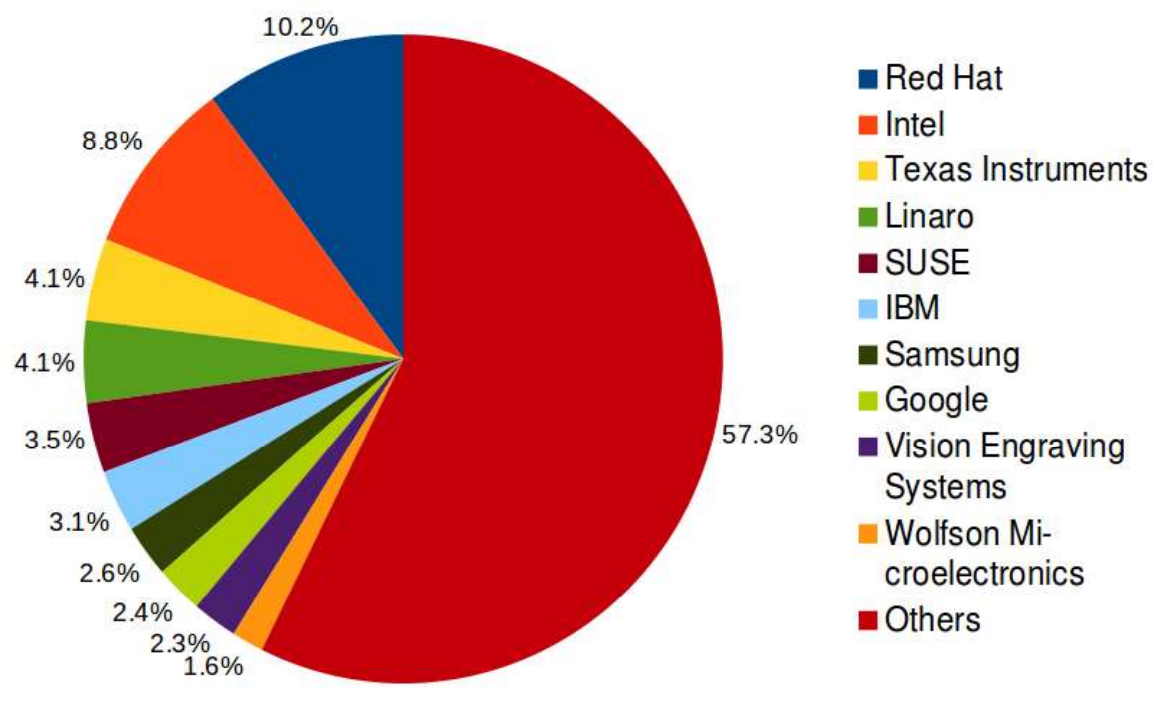


Why Linux?

Why Linux?

- **Why Linux ?**
  - It is Open source :)
  - Linux is everywhere: smart phones, tablets, T.Vs, Cars, space stations
  - Linux is present in highly critical applications such as Japan’s bullet trains, traffic control, Stock Exchange,many air traffic control systems or control of nuclear reactors.

Why Linux?



the top-10 corporate sponsors of Linux kernel development, in terms of total commit counts from their employees, as of year 2013
http://xmodulo.com/interesting-facts-linux.html

Getting Started

Installation

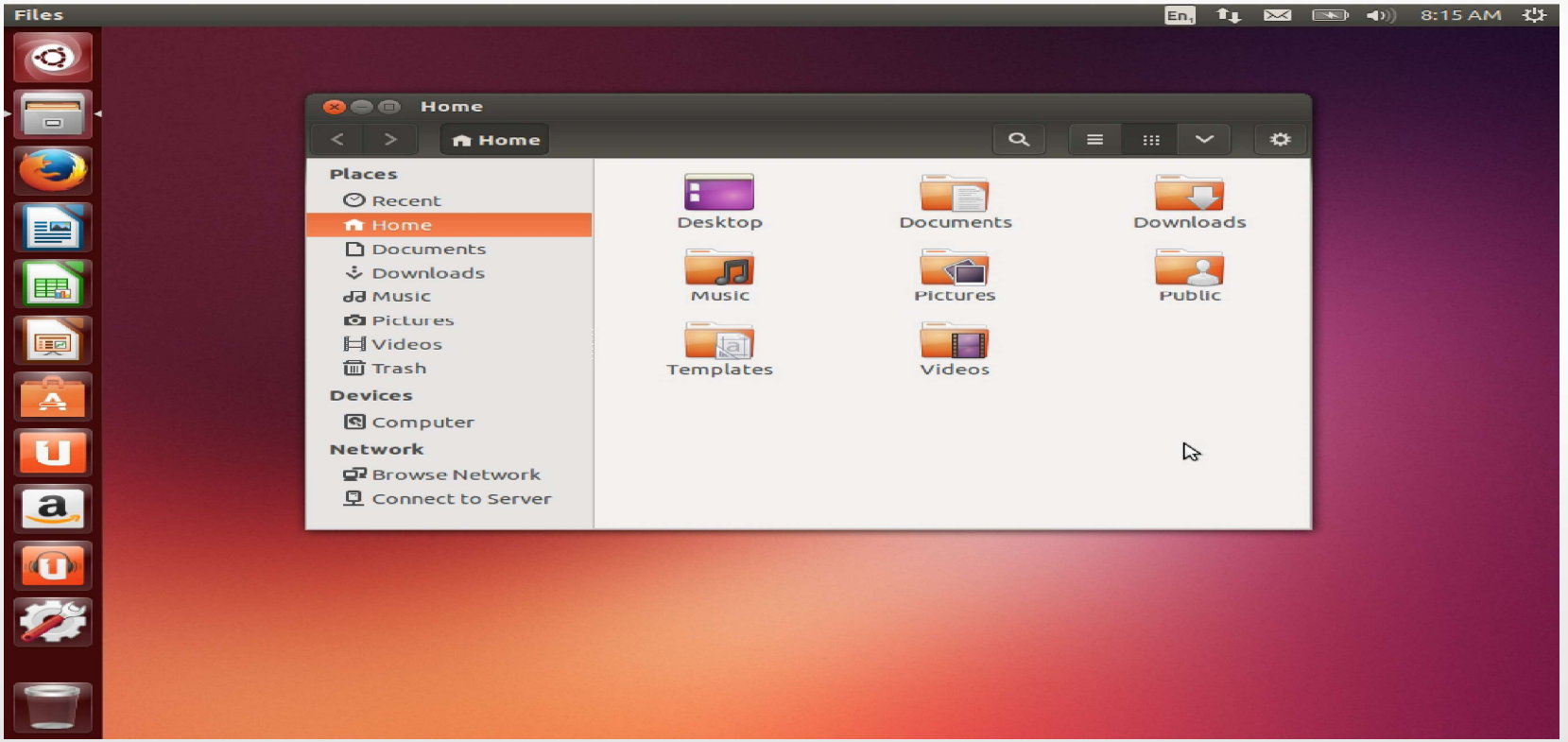
- Ubuntu Desktop Edition
  - 700 MHz processor
  - 512 MiB RAM
  - 5 GB of hard-drive space
  - VGA capable of 1024x768 screen resolution
- Ubuntu Server (CLI) Installation
  - 300 MHz x86 processor
  - 192 MiB of RAM
  - 1 GB of disk space
  - Graphics card capable of 640x480

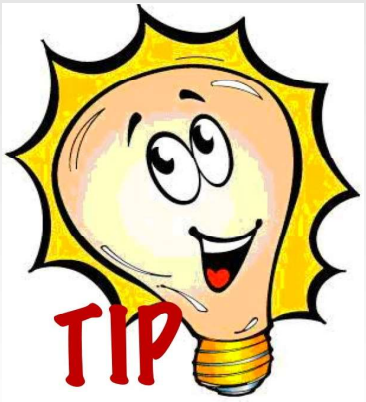
Types of Installation

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

Getting Started

Unity





If you truly wish to master a skill,
nothing beats hands-on experience

SOOOO :)
Let's Start !

Getting Started

- **The Launcher**
  - Area in the Unity Desktop where you have access to certain actions
  - One of the launcher's main functions is its search bar that you can find in the main menu and in the Applications and Files & Folders sections.
- **Applet**
  - A small interactive application that resides within the panel for example the volume control.
- **Workspace**
  - A discrete area in the Unity Desktop in which you can work.

If you truly wish to master a skill,
nothing beats hands-on experience

SOOOO :)
Let's Start !

Start with Terminal

Start with Terminal

- The command line is provides by a program called shell.
- Using the command line: Commands are entered in a terminal at the shell prompt.
  - The default prompt is the login name of the current user, the hostname, the current directory between square brackets, followed by $

    [msabagh@localhost Desktop]$
  - “$” is replaced by “#” in case of root

Start with Terminal

- Commands have the following syntax:

  command [options] [arguments]

  - Each item is separated by a space.
  - Options modify the command's behaviour.
  - Arguments are files name or other information needed by the command.

Start with Terminal

- Useful Bash Features:
  - Tab completion allow you to quickly complete commands and file names:

    [msabagh@localhost ~]$ pas<Tab>

    ```
    passwd    paste    pasuspender
    ```

  - **[msabagh@localhost ~]$ passwd**
- Separate commands with semicolon (;)
- “--help” option print a description about the command

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

cal ;uname

Cal 5 2002; date; uname

How To Fish

How to fish?

- Google
- Ubuntu community

  http://community.ubuntu.com/
- Local documentation

How to fish?

Local documentation:

- Unity Help (a collection of graphical hypertext books). To access Unity Help Browser:
  - → Press 'F1' or select Applications → Documentation → Help
- Additional documents are stored in the /usr/share/doc directory
- Built-in Linux System Manual (man pages for commands, configuration files and programming calls) using command line type man

If you truly wish to master a skill,
nothing beats hands-on experience

SOOOO :)
Let's Start !

How to fish?

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

How to fish?

- Bugs
  - Known bugs and errors
- See also
  - Other commands related to this one

Manual Sections

1.User commands

2.System calls

3.C Library Functions

4.Devices

5.File formats and protocols

6.Games

7.Miscellanea

8.System Administration tools and Deamons

How to fish?

man –k keyword

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

whatis command

Shows the commands one line description

-help Option

Another way to get help about a command.

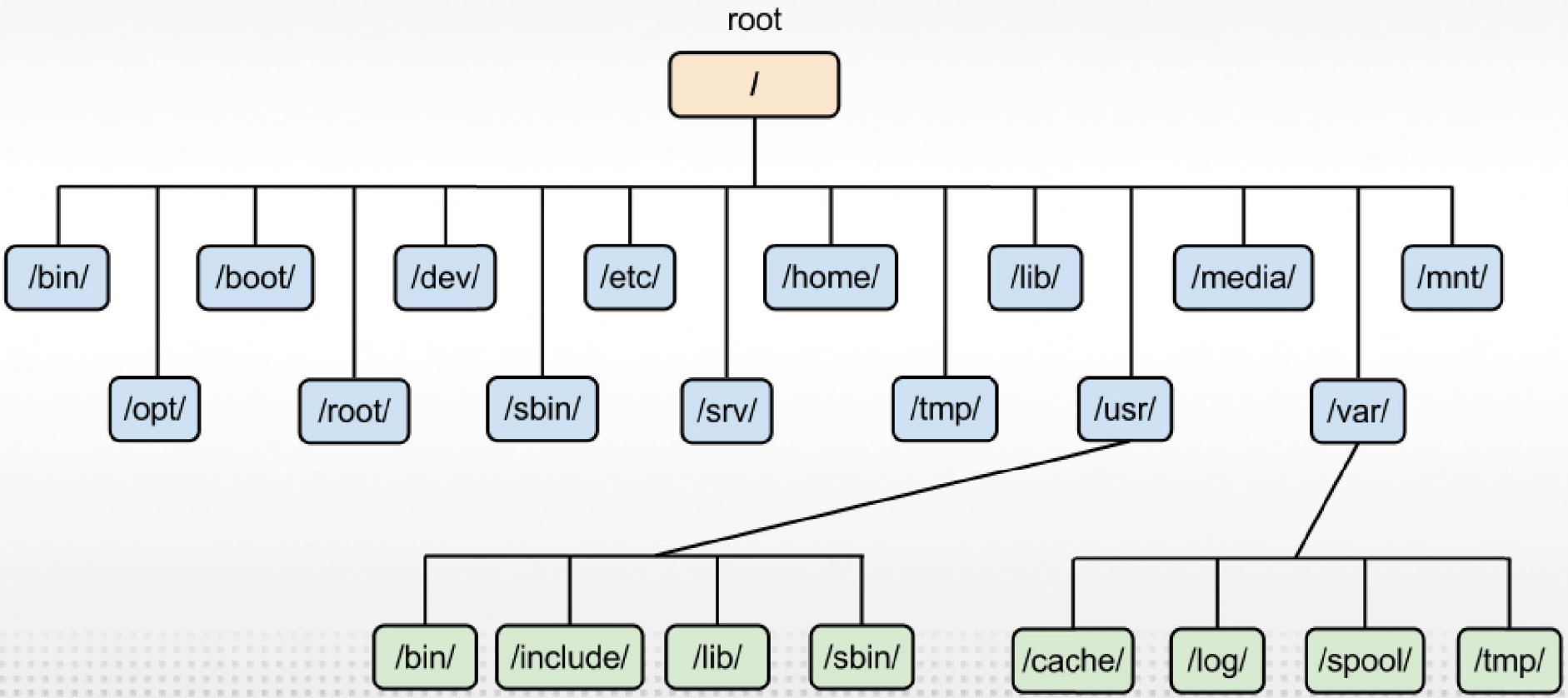
Introduction to Directories

- 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

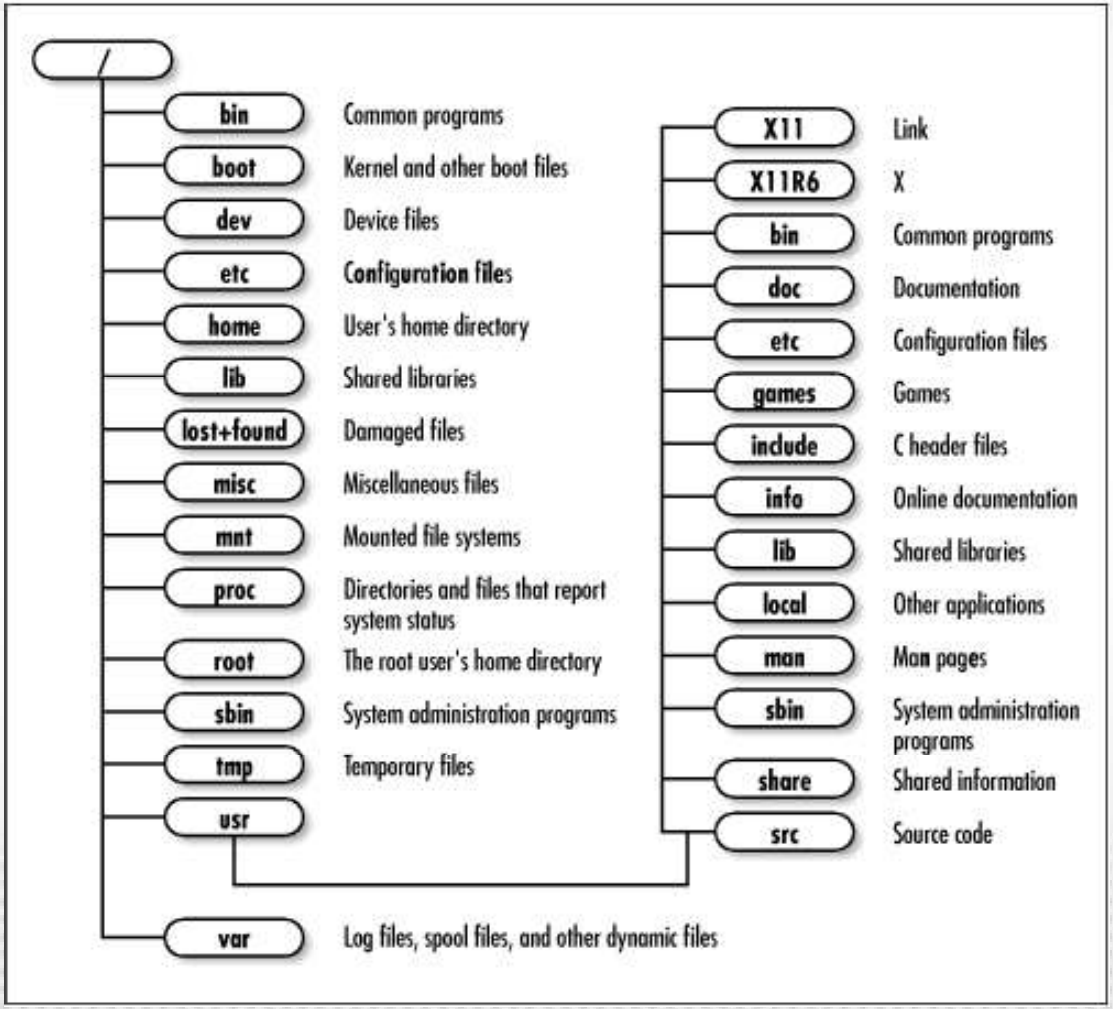
/home/guest

How Directories Work ?



How Directories Work ?

- 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
```

File Naming

- File names may be up to 255 characters.
- 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 and Directory 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 and Directory 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 and Directory Manipulation

- To create files

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

- To create directories

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

File and Directory Manipulation

- To remove files

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

- To remove directories

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

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

Listing Directory Contents


```
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 –a dir1
.        .f1  f1
..       .f2   f2
```

How Directories Work ?

- Pathnames
  - Absolute pathname
  - Relative pathname

Files names beginning with '.' are hidden files. To show hidden files select :

- Press <Ctrl>+H
- From view options > Show Hidden Files.

Changing Directories

- To move from directory to directory on the system

```
cd /home/user1/work

cd ..

cd ~

cd -
```

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
```

Metacharacters

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

Examples

```
ls file?
file4 file1 file2
```

- 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
ls *a*
abm bam bat battle
```

Metacharacters


```
ls  ???

abm bam bat

ls  ?a?

bam bat

ls  ?a*

bam bat battle
```

Metacharacters


```
ls [ab]*
```

```
abm bam bat battle
```

Contents

- Vi text editor
- User and group administration
- Files Permissions

The vi text editor Cont'd

- VI editor is an interactive editor that you can use to create and modify test files.

- It is used when the desktop environment window system is not available.

Fundamentals VI Operations

- VI three basic modes
  - **Command mode**
    - Default mode
    - Perform commands to delete, copy, …

Fundamentals VI Operations

- VI three basic modes
  - **Edit mode**
    - Enter text into the file
  - **Last line mode**
    - Advanced editing commands
    - To access it, enter a colon (:) while in the command mode

Fundamentals VI Operations

- To enter edit mode
  - **i** Inserts text before the cursor
  - **o** Opens a new blank line below the cursor
  - **a** Appends text after the cursor
- After editing Press **esc** to enter command mode

Fundamentals VI Operations

- The syntax of vi command
  - **vi**
  - **vi** filename
  - **vi** options filename
- To recover a file
  - **vi –r** filename

Manipulating Files Within VI

- **Viewing files in Read-only mode**
  - view filename
    - Perform the **:q** command exit
- **Inserting and appending text**
  - **A** append text at the end of the line
  - **I** insert text at the beginning of the line
  - **O** opens a new line above the cursor

Manipulating Files Within VI

- **Moving the cursor within the vi**
  - **h**, left arrow, or backspace: left one character
  - **j** or down arrow: down one line
  - **k** or up arrow: up one line
  - **l**, right arrow or space: right one character

Manipulating Files Within VI

- **Moving the cursor within the vi (cont.)**
  - **w** forward one word
  - **b** back one word
  - **e** to the end of the current word
  - **0 (zero)** to the beginning of the line

Manipulating Files Within VI

- **Moving the cursor within the vi (cont.)**
  - **G** Goes to the last line of the file
  - **:n** Goes to Line n

Manipulating Files Within VI

- **Substitute and delete text**
  - **x** Deletes a character at the cursor.
  - **dw** Deletes a word or part of the word to the right of the cursor.
  - **dd** Deletes the line containing the cursor.
  - **D** Deletes the line from the cursor to the right end of the line.
  - **:n,nd** Deletes Lines n through n (in last line mode)

Manipulating Files Within VI

- **Search and replace**
  - **/string** Searches forward for the string.
  - **?string** Searches backward for the string.
  - **n** Searches for the next occurrence of the string.
  - **N** Searches for the previous occurrence of the string.
  - **:%s/old/new/g** Searches for the old string and replaces it with the new string globally. (In the last line mode)

Manipulating Files Within VI

Copy and paste

- **yy** Yank a copy of a line.
- **p** Put yanked text under the line containing the cursor.
- **P** (upper Case) Put yanked text before the line containing the cursor.
- :**n,n co n** Copy Lines n though n and puts them after Line n. (Last Line Mode)
- **:n,n m n** Move Lines n through n to Line n. (Last Line Mode)

Manipulating Files Within VI

- **Save and quit**
  - **:w** save the file
  - **:w** new_file save as new file
  - :**wq, :x** save and quit
  - **:q!** quit without saving

Manipulating Files Within VI

- **Customizing vi session**
  - :**set nu, :set nonu** show and hide line numbers
  - :**set ic, :set noic** ignore or be case sensitive
  - **:set showmode, :set noshowmode** display or turn off mode

Editing Files with gedit

- The gedit text editor is a graphical tool for editing text files.

- The gedit window is launched by selecting:
  - Search menu → gedit

Users and Groups databases

- The `/etc/passwd` file

```
username:x:uid:gid:comment:home-directory:login-shell
```

Users and Groups databases

- The `/etc/shadow` file

```
username:encrypted passwd:last
changed:min:max:warn:??:expire:future-use
```

Users and Groups databases

- The `/etc/group` file

```
groupname:x:gid:comma-separated list of group members
```

- The `/etc/gshadow` file ???

Adding a new user account


```
# useradd username
```

- The `useradd` command populates user home directories from the `/etc/skel` directory.

- Adding multiple user accounts

```
# newusers filename
```

User Password


```
# passwd username

$ passwd
```

Password Aging Policies

- The `chage` command sets up password aging

```
# chage [options] username
```

- Options
  - -m: to change the min number of days between password changes
  - -M: to change the max number of days between password changes
  - -E date: change the expiration date for the account
  - -W: change the number of days to start warning before a password change will be required

Modifying user accounts

- To change a user's account information, you can:
  - Edit the `/etc/passwd` or `/etc/shadow` files manually
  - Use the usermod & `chage` command discussed later

Modifying user accounts

- To change a user's account information, you can:
  - Use the `usermod` command:
  - `usermod [options] username`
    - Useful options
      - To changes the login name use -l <login name>
      - To lock the password use -L
      - To unlock the password use -U

Deleting a user account

- To delete a user account you can
  - Manually remove the user from
    - `/etc/passwd` file
    - `/etc/shadow` file
    - `/etc/group` file
    - remove the user's home directory (`/home/username`)
    - and mail spool file (`/var/spool/mail/username`)
  - Use the `userdel` command.

```
# userdel [-r] username
```

Switching Accounts


```
# su [username]
```

User private group scheme

- Ubuntu assigning user a primary group for which they are the sole members.
- This "private" primary group has the same name as the user's username

Managing Groups

- Creating New Group

```
# groupadd groupname
```

- Modifying an Existing Group

```
# groupmod [options] groupname
```

- Deleting a Certain Group

```
# groupdel groupname
```

Managing Groups cont'd

- You can use the `gpasswd` command to define
- gpasswd -a USERNAME GROUPNAME
  - Add members to Group
  - create or change group passwords

Changing Active Group

- To display the groups you are member in use `groups` command

```
groups
other root bin sys adm uucp mail tty lp
```

- To switch between groups you are member in, use `newgrp` command.

```
newgrp group
```

The whoami command

- After switching into several users, it is a sever issue to know your current (effective)

user

```
whoami
```

```
Root
```

- `id`

uid=101(user1) gid=100(user1)groups=101(user1)

```
id user2
```

uid=500(user2)gid=500(user2) groups=500 (user2)

The id command

- Displays
  - Effective user id
  - Effective user name
  - Effective group id
  - Effective group name
- Examples
- `id`

uid=101(user1) gid=100(user1)groups=101(user1)

```
id user2
```

uid=500(user2)gid=500(user2) groups=500 (user2)

File Ownership and Permissions

- Every file and directory has both **user** and **group** ownership. A newly-created file will be owned by:
  - The user who creates it
  - That user's primary group

File Ownership and Permissions

- File ownership can be changed using `chown` command.
- Example

```
# chown user1 file1
# chown user1:group1 file1
# chown :group1 file1
```

Security Scheme

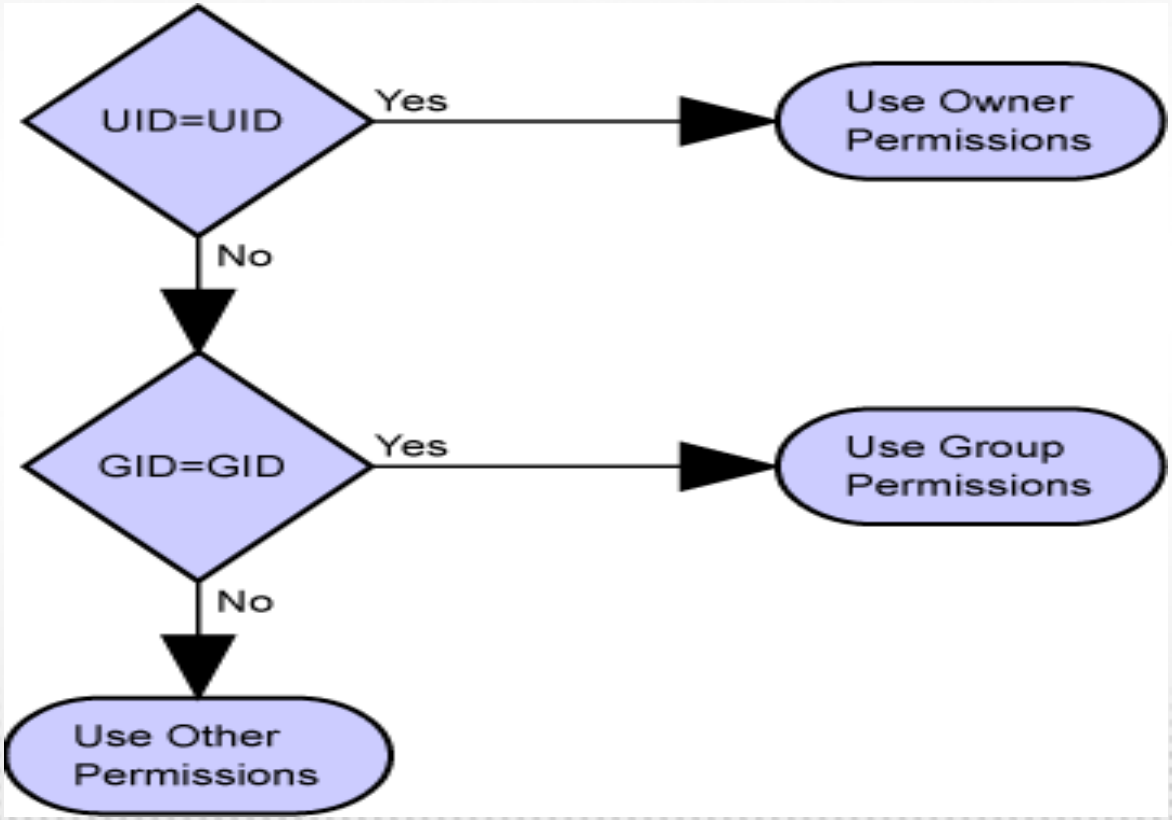
- Each file has an owner and assigned to a group.
- Linux allows users to set permissions on files and directories to protect them.
- Permissions are assigned to
  - File owner
  - Members of the group the file assigned to
  - All other users
- Permissions can only be changed by the owner and root

Permission Notations

Permission	Access for a File	Access for a Directory
Read	You can display file contents and copy the file.	You can list the directory contents with the ls command
Write	You can modify the file contents.	If you also have execute access, you can add and delete files in the directory.
Execute	You can execute the file if it is an executable. You can execute a shell script if you also have read and execute permissions.	You can use the cd command to access the directory. If you also have read access, you can run the ls -l command on the directory to list contents.

Determining Permissions

The most specific permissions apply



Changing the Permissions

`chmod permission filename`

- Permissions are specified in either
  - Symbolic mode
    - Who
      - u: Owner permissions
      - g: Group permissions
      - o: Other permissions
      - a: all permissions

Changing the Permissions

- Permissions are specified in either
  - Symbolic mode
    - Operator
      - + Add permissions
      - - Remove permissions
      - = Assign permissions absolutely
    - Permissions
      - r: read
      - w: write
      - x: execute

Changing the Permissions

- Permissions are specified in either
  - Octal mode
    - 4 read
    - 2 write
    - 1 execute

Examples

ls -l file1

```
-rw-r--r-- 1 user1 staff 1319 Mar 22 14:51 file1
```

chmod o-r file1

ls -l file1

```
-rw-r----- 1 user1 staff 1319 Mar 22 14:51 file1
```

chmod g-r file1

ls -l file1

```
-rw------- 1 user1 staff 1319 Mar 22 14:51 file1
```

Examples Cont'd

chmod u+x,go+r file1

ls -l file1

```
-rwxr--r-- 1 user1 staff 1319 Mar 22 14:51 file1
```

chmod a=rw file1

ls -l file1

```
-rw-rw-rw- 1 user1 staff 1319 Mar 22 14:51 file1
```

chmod 555 file1

ls -l file1

```
-r-xr-xr-x 1 user1 staff 1319 Mar 22 14:51 file1
```

Examples Cont'd

chmod 775 file1

ls -l file1

```
-rwxrwxr-x 1 user1 staff 1319 Mar 22 14:51 file1
```

chmod 755 file1

ls -l file1

```
-rwxr-xr-x 1 user1 staff 1319 Mar 22 14:51 file1
```

Agenda

- Network Interfaces
- Environment Variables
- String Processing
- Virtual Consoles

Special Permissions

Special Permission	Effect on Directories
g+s	Files newly created in the directory have their group owner set to match the group owner of the directory
o+t	Users with write permission on the directory can only remove files that they own

Special Permissions

- Example

```
# chmod g+s directory1
# ls –l
drwxr-sr-x ...

# chmod o+t directory1
# ls –l
drwxrwxrwt ...
```

Network Interfaces

- Interface names
  - `eth0`
  - `eth1`
  - `eth2`
- To list the interface names for all NICs on your computer(view MAC address)
  - `Ifconfig`
- To configure IP addresses on network interfaces

```
sudo ifconfig eth0 192.168.1.14
```

Network Interfaces Commands

- To bring up or down a network interface

```
# ifdown eth0

# ifup eth0
```

Configuration Utilities cont'd

- To specify your DNS server manually in `/etc/resolv.conf`

```
nameserver 4.2.2.1
nameserver 4.2.2.2
```

Setting/Changing The hostname

- The `hostname` command allows you to directly query, or set, the hostname from the command line.

```
Hostname

myserver
```

- `To change it`

```
Hostname -b NewName
```

Use network diagnostic tools

- `host hotmail.com`

  hotmail.com has address 64.4.20.169

- The `ping` command is a network packet loss and latency measurement tool

- The `traceroute` command will attempt to show the network packets' router path between the local system and a remote system.

Client-side DNS configuration

- Local name resolution can eliminate the need for DNS look-ups by modifying the `/etc/hosts`

- `192.168.5.55 mywebapp.mydomain.com`

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 value.
- The value of an environmental variable can for example be the location of all executable files in the file system, the default editor that should be used, or the system locale settings

Environment Variables

- $HOME
  - Complete path of the user home directory
  - Example
    - `mkdir $HOME/file1`
- $PATH
  - A colon-separated list of directories used by the shell to look for executable program names
  - Example
    - `echo $PATH`

/usr/bin:/bin:/usr/local/java/bin

Environment Variables Cont'd

- $PWD
  - The user current working directory
- $SHELL
  - Path name of the login shell
- $USER
  - Currently logged in user
- $HOSTNAME
  - Name of the computer

Viewing variable contents

- The shell assumes whatever follows the dollar sign ($) in the command line is a variable and substitutes its value
  - `echo $HOME`

/home/user

- To view the contents of all variables by running the `env` command

Command Alias


```
alias l.='ls .* ‘

alias ll='ls -l ‘
```

- Type `alias` at the terminal to **see all set aliases**
- **Remove aliases**

`unalias` command

- **Bypass aliases**

```
alias ls='ls -AF'

/usr/bin/ls

\ls
```

Commands History

- bash stores a history of commands you have entered so that you can recall them later.
- The history is stored in the user's home directory and is called `.bash_history` by default.

Start-up Scripts

Start-up scripts are scripts of commands executed at login. They are used to:

- Set up the environment
- Establish commonly used aliases
- Run programs

Global Initialization Files

- ~/.bashrc

By default this file will be executed in each and every invocation of bash or logging the user as well as while logging in to the graphical environment.

- `/etc/bash.bashrc`

This is the system-wide version of the ~/.bashrc file. By default this file is executed whenever a user enters a shell or the desktop environment.

Initialization Files

•/etc/environment

It is not a script file, but rather consists of assignment expressions, one per line. Specifically, this file stores the system-wide locale and path settings.

●/etc/profile.d/*.sh

This is the system-wide version of the ~/.bashrc file. By default all files in this directory executed whenever a user enters a shell or the desktop environment .

String Processing

- Use the `wc` and the `diff` commands to gather word file statistics and compare two files
- Search strings for patterns using the `grep` command
- Move and delete data using `cut` command
- Organize data using the `sort` command

The wc command

- The `wc` command displays the number of characters, words, and lines in a specified file.
- The syntax for the wc command is:
  - `wc [option] [filename]`
- The `wc` command is often used when differentiating between two versions of a file.

The wc command Cont'd

- Word-count command options

Option	Meanings
-c	Count the number of characters only
-l	Count the number of lines only
-w	Counts the number of words only

The wc command

- For example,

```
$ wc story.txt

39 237 1901 story.txt
```

The diff command

- The diff command is also used to compare the contents of two files for differences.

  If you upgrade a utility and want to see how the new configuration files differ from the old, use the diff command

```
diff /etc/named.conf.rpm.new /etc/named.conf
```

Searching for Content in Files

- Displays the lines of its input that match a pattern given as an argument
- `grep options regular-expression filename(s)`

Option	Description
-I	Ignore case sensitive
-l	List files name
-n	Precedes each line with relative line number in the file
-c	Counts the line that contains the pattern

The cut command

- cut command cuts fields or columns of text from standard input or the named file and displays the result to standard output
  - cut option[s] [filename]
    - -f specifies field or column.
    - -d specifies field delimiter (default is TAB).
  - cut -f3 -d: /etc/passwd

Contents

- Processes, priorities and signals Concepts.
- Redirection & Piping
- Apt-get utility
- Search
- Archiving

Processes

- Every program you run creates a process. For example
  - Shell
  - Command
  - An application

Processes

- System starts processes called **daemons** which are processes that run in the background and provide services

Every processes has a **PID**

- Every processes has a PID, positive integer between 2 and 32,768, The number 1 is typically reserved for the special init process
- When a process creates another, the first is the **parent** of the new process. The new process is called the **child process**. Parent waits for her child to finish

Process Priority

- Only process at a time may be executed on the CPU.
- Every process which is ready to run has a scheduling priority.
- The Linux process divides CPU time into time slices, in which each process will get a turn to run, higher priority processes first.
- User can affect the priority by setting the niceness value for a process

Process Priority Cont'd

- Niceness values range from -20 to +19, which indicates how much of a bonus or penalty to assign to the priority of the process.

- Most processes run with a niceness value of 0 (no change).

Process Priority Cont'd

- Smaller numbers are higher priority. Processes with a higher priority will run first in each time slice, and will run longer before its turn to run ends.

- Users can adjust this value down as far as +19 but can not increase it. Root can increase the priority of a process as high as -20

Adjusting Priority of a Process

- Adjusting process priority at invocation time

```
nice [-n adjustment] command
```

```
nice -n 10 makewhatis
```

- Adjusting the priority of a running process

```
renice priority [[-p] pid ...] [[-g] group ...] [[-u] user ...]
```

Signals

- A signal is a message sent to a process to perform a certain action
- Signals are identified by a signal number and a signal name, and has an associated action.
  - SIGTERM → 15
  - SIGKILL → 9

Viewing Processes

- Jobs show process in current terminal

Viewing Processes

- Process status command

```
ps option(s)
```

  - Output
    - PID
    - TTY -> terminal identifier
    - Execution time
    - Command name
  - Options
    - -e: all system processes
    - -f: full information
    - -u uid: display processes of that user.

Viewing Processes

Viewing processes with `top` utility

Searching for a Process

- Using `pgrep` command

```
pgrep option(s) pattern
```

```
$pgrep lp
```

- Options

-x: exact match

-u uid: processes for a specific user

-l: display the name with pid

Sending a Signal to a Process

- Using kill command
  - Default signal 15

```
kill [-signal] PIDs
```

```
Kill -l5  10345
```

- Examples

```
$pgrep -l mail
215 sendmail
12047 dtmail
```

Sending a Signal to a Process


```
$kill 12047


$pgrep -l mail
215 sendmail
```

- Using `pkill` command

```
pkill [-signal] process_name
```

- Example

```
$pkill -9 dtmail
```

Examples


```
sleep 500 &
[1] 3028
[1] + Done
jobs
[1] + Running     sleep 500&
fg 1
sleep 500
sleep 500
Ctrl+Z [1] + Stopped (SIGTSTP) sleep 500
jobs
[1] + Stopped (SIGTSTP) sleep 500
```

Examples Cont'd


```
bg %1
[1] sleep 500&
jobs
[1] + Running sleep 500&
kill -STOP %1
jobs
[1] + Stopped (SIGSTOP) sleep 500&
kill %1
[1] + Terminated sleep 500&
jobs
```

Standard Input and Output

- **Standard input**
  - Refers to the data source from which data is input to a command
  - Typically the keyboard
- **Standard output**
  - Refer to data destination to which data from the command is written
  - Typically the screen

Standard Input and Output

- **Standard error**
  - Refer to the output destination for the errors and messages generated by the command
  - Typically the screen also

Redirecting Input and Output

- `Command > fname`
- `Command >> fname`
- `Command < fname`

— Example

- `$ find /etc –name passwd > findresult`
- `$ ls –l /etc >> findresult`
- `Mail < file1`

Redirecting Standard Error

- Standard error is redirected to a file using the regular output redirection operator, but you must place a 2 in front of the operator (2>).
- Example
  - $ find / -name passwd 2> errs
  - $ find / -name passwd 2> errs > results

Using Pipe to connect Processes

- The pipe
  - A pipe (|) is used to send the output of one command as the input to another
  - The most common use of a pipe is to take a command that's output might go on for pages (such as cat or ls -l) and feed it through more.
  - Example
    - `$ ls –lR / | more`

APT-get (Advanced Packaging Tool)

- Ubuntu's package management system is derived from the same system used by the Debian GNU/Linux distribution.
- The package files contain all of the necessary files, meta-data, and instructions to implement a particular functionality or software application on your Ubuntu computer.
- The software management tools in Ubuntu will check dependencies automatically.

APT-Get

- The apt-get command is a powerful command-line tool used to work with Ubuntu's Advanced Packaging Tool (APT) performing such functions as
  - Installation of new software packages
  - Upgrade of existing software packages
  - Updating of the package list index
  - And even upgrading the entire Ubuntu system.

APT-Get Cont'd

- Install a Package

```
sudo apt-get install ksh
```

- Remove a Package

```
sudo apt-get remove ksh
sudo apt-get purge nmap
```

- * You may specify multiple packages to be installed or removed, separated by spaces.

APT-Get Cont'd

- Update the Package Index
  - Update is used to resynchronize the package index files from their sources.
  - The indexes of available packages are fetched from the specifies location(s) in /etc/apt/sources.list.
  - An update should always be performed before upgrade.

```
sudo apt-get update
```

APT-Get Cont'd

- Upgrade package
  - Install the newest versions of all packages currently installed on the system from the sources found in /etc/apt/sources.list.
  - An update must be performed first so that apt-get knows the new versions of packages available.

```
sudo apt-get upgrade
```

Finding Files with locate

- The locate command searches through a pre-built database containing the contents of your filesystem at the time the database was last updated.
- The locate database is built by using the `updatedb` command.
- Example
  - `locate passwd`

Locating Files with find

- The find command searches the live filesystem.
- find is slower than locate, causes more of a load on the system, but more powerful than locate.
- You are also limited by your own permissions.

Locating Files with find Cont'd

Expression	Definition
-name filename	Finds files matching the specified filename. Metacharacters are acceptable if placed inside " ".
-size [+|-]n	Finds files that are larger than +n, smaller than -n, or exactly n. The n represents 512-byte blocks.
-atime [+|-]n	Finds files that have been accessed more than +n days, less than -n days, or exactly n days.
-mtime [+|-]n	Finds files that have been modified more than +n days ago, less than -n days ago, or exactly n days ago.
-user loginID	Finds all files that are owned by the loginID name.
-type	Finds a file type, for example, f (file) or d (directory).
-perm	Finds files that have certain access permission bits

The sort command

- The sort command sorts text data after accepting it from either a file or the output of another command.

- Examples:
  - sort –t : -k1 /etc/passwd
  - sort –t : –k3 /etc/passwd
  - sort –t : -n –k3 –o passwd_sorted /etc/passwd

The tee command

- The `tee` command reads from the standard input and writes to the standard output and a file
- Example
  - `$ ls –lR / | tee fname | more`

Introduction To Archiving

- To safeguard your files and directories, you can create a copy, or archive, of the files and directories on a removable medium, such as a cartridge tape. You can use the archived copies to retrieve lost, deleted, or damaged files.

Archiving Files

- `tar` command archives files to and extracts files from a single file called a tar file.
- The default device for a tar file is a magnetic tape device.
- tar functions archivefile filenames
  - Function
    - c: create a new tar file
    - t: list table of content
    - x: extracts files from the tar command
    - f: specify the archive file
    - v: verbose mode

Archiving Files

- Example
- `tar cvf file.tar file1 file2 file3`

file1

file2

Archiving Files Cont'd

- Viewing an archive
  - tar tf file.tar
    - file1
    - file2
    - file3
- Extracting files from archive
  - tar xvf file.tar
    - file1
    - file2
    - file3

Compress Command

- Compression reduces a text file by 50 percent to 60 percent.
- compress [-v] filename
- Compress command replaces the original file with a new file that has a .Z extension.
- Example
  - compress -v files.tar
    - files.tar: Compression: 70.20% --
    - replaced with files.tar.Z

zcat Command

- zcat filename.Z

- Example
  - zcat file1.Z

uncompress Command

- uncompress options filename
- Example
  - uncompress -v files.tar.Z
    - files.tar.Z:-- replaced with files.tar

bzip2 Command

- The bzip2 command reduces the size of files.
- The original file is replaced by a file with the same name and a .bz2 extension.
- bzip2 [-v] filenames
- Examples:
  - bzip2 file1 file2 file3 file4
  - ls *.bz2
    - file1.bz2 file2.bz2 file3.bz2 file4.bz2

bzip2 Command Cont'd

- Restoring bzip2 file using the bunzip2 command
- Example
  - bunzip2 file1.bz2

bzcat Command

- bzcat command display the content of files compressed by bzip2

  —bzcat filename.bz2

Checking Free Space

- The `df` command displays number of free disk blocks and files.

```
df [-h]
```

- Example

```
df –h
Filesystem size used avail capacity Mounted on
/dev/hda0  15G  976M  14G   6%       /
```

Checking Free Space

- The du command display the total sum of space allocated to all files hierarchy rooted in the directory specified.

```
du [-sh] [dir…]
```

- Example

```
du –sh
   14K
```

System Shutdown

- It only requires reboot or shutdown when you need to
  - Add or remove hardware
  - Upgrade to a new version of Ubuntu
  - Or upgrade your kernel

  - `shutdown time`
  - `poweroff`
  - `Init 0`
  - `halt`

System reboot

- `shutdown -r`
- `reboot`
- `init 6`
- Press `CTRL+ALT+DEL`

Virtual Consoles

- Accessed with Ctrl-Alt-F_key
- Consoles 1-6 accept logins
- X server starts on the console F1
- Console from F2 to F6

Inode

- Linux see all files as numbers called “inodes”, or index nodes.

- Within each each filesystem is an inode table, in which all of the used inodes are mapped to particular files.

Inode Cont'd

- The information stored in this table for each entry includes the following:

1. The type of file
2. The file's permissions
3. The number of links
4. The file owner's user ID
5. The group owner's GID
6. When the file was last changed
7. When the file was last accessed
8. Where the file is on the media

Inode Cont'd

- To view inode number of a file
  - `ls -i fname`
    - 10978 fname

File Manipulation and Inodes

- The cp command
  - Allocates a new inode number for the copy, placing a new entry in the inode table
  - Creates a directory entry, referencing the file name to the inode number within that directory

File Manipulation and Inodes Cont'd

- Example
  - `ls -i f1`

1196100 f1

  - `cp f1 f2`
  - `ls -i f1 f2`

1196100 f1

1196463 f2

File Manipulation and Inodes Cont'd

- Inodes and the mv command
  - If the destination is on the same file system as the source:
  - mv creates a new directory entry with the new file name
- Example
  - `ls -i f1`

1196100 f1

  - `mv f1 f2`
  - `ls -i f2`

1196100 f1

Utilizing Links

- Soft Link (Symbolic Link)
  - The content of this entry is the path to the original file.
  - This allows you to use symbolic links across partition boundaries.
  - If you delete the original file, you end up with an “orphaned link”

Utilizing links to make shortcuts Cont'd

- Example

```
ls -l testfile
```

-rw-rw-r-- 1 user user 12 Mar 12 03:50 testfile

```
ln -s testfile testlink
```

```
ls -l testfile testlink
```

-rw-rw-r-- 1 user user 12 Mar 12 03:50 testfile

lrwxrwxrwx 1 user user 8 Mar 12 09:50 testlink → testfile

Utilizing links to make shortcuts Cont'd

- Hard Links
  - Instead of creating a new file, the new link (a new directory entry) is added to the appropriate directory file name listing, referencing the exact inode as the original file. Thus, the file only exists once, but in two places.
  - In the inode table, the link count is incremented.
  - Every filesystem has inodes that start counting from zero. A hard link cannot reach across partition boundaries. It can only exist within a single partition or media.

Utilizing links to make shortcuts Cont'd

- Example
  - `ls -l testfile`

-rw-rw-r-- 1 user user 12 Mar 12 03:50 testfile

  - `ln testfile testlink`
  - `ls -l testfile testlink`

-rw-rw-r-- 2 user user 12 Mar 12 03:50 testfile

-rw-rw-r-- 2 user user 12 Mar 12 09:50 testlink