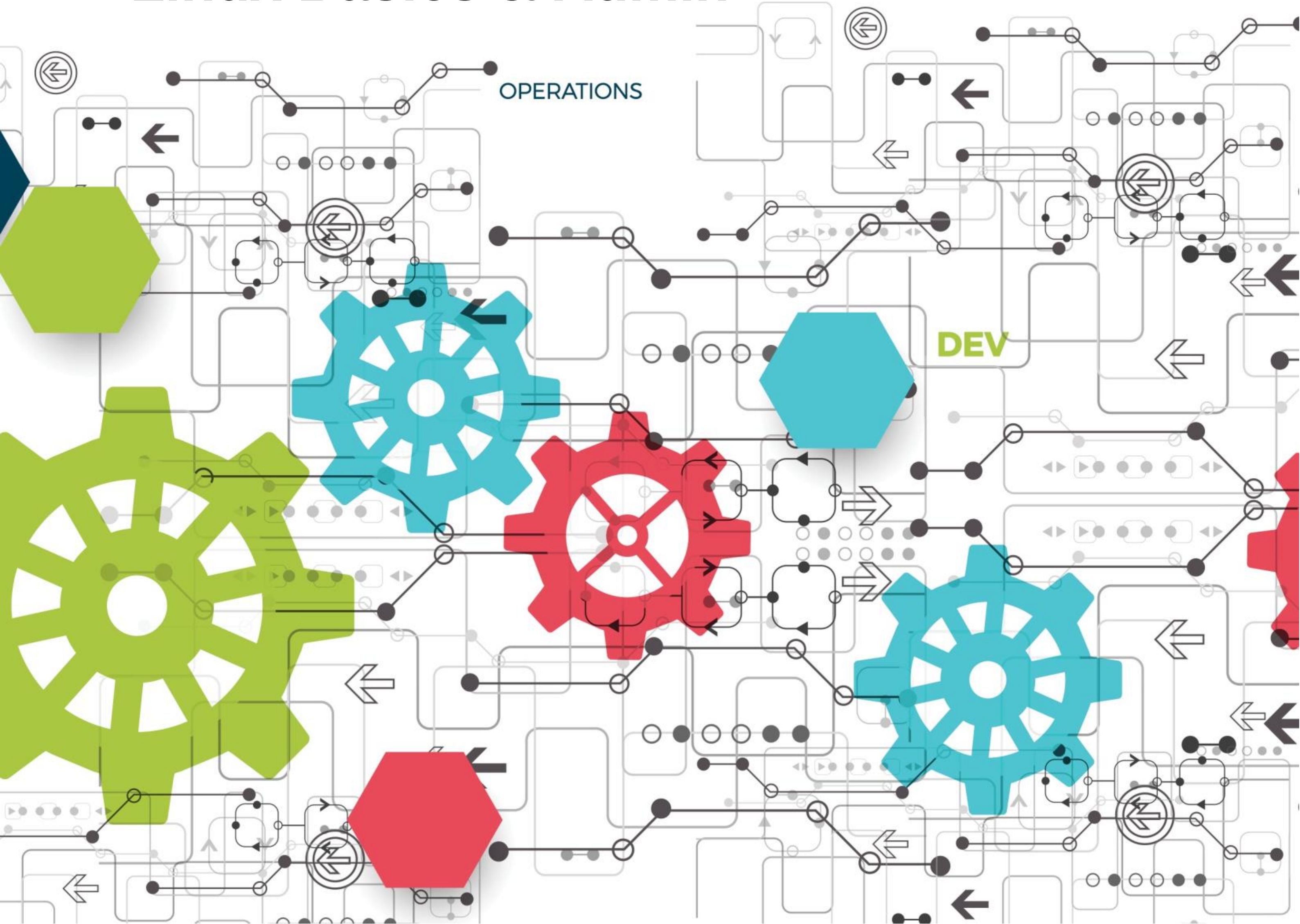
Xebia

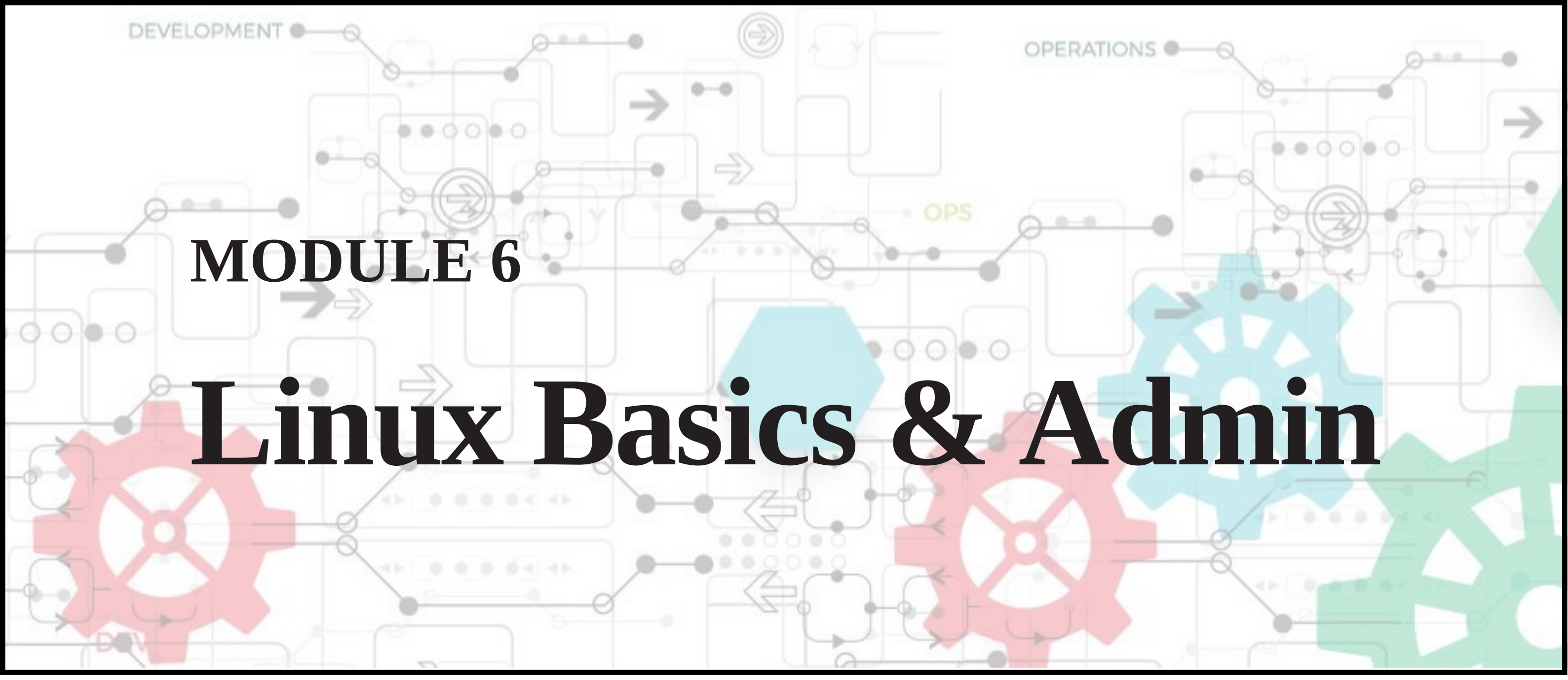
B.Tech Computer Science and Engineering in DevOps

DevOps Overview

MODULE 4

Linux Basics & Admin





MODULE 6

Linux Basics & Admin

Facilitator Notes:

Introduce the module to the participants and tell them that you will be talking about Linux, Relation & Difference with respect to DevOps, Linux Basic Command Utilities, Linux Administration, Environment Variables, Networking, Linux Server Installation, RPM and YUM Installation

You will be discussing about Linux, the different types of Linux distributions, Linux Basic Command Utilities, Linux Administration, Environment Variables, Networking, Linux Installation, RPM and YUM Installation.

Module Objectives

At the end of this module, you will be able to:

- List different types of Linux flavors
- Relate DevOps and Linux
- Choose Linux Distributions
- Understand Basic Linux Commands
- Describe Linux installation

Facilitator Notes:

Explain the module objectives to the participants.

Module Topics

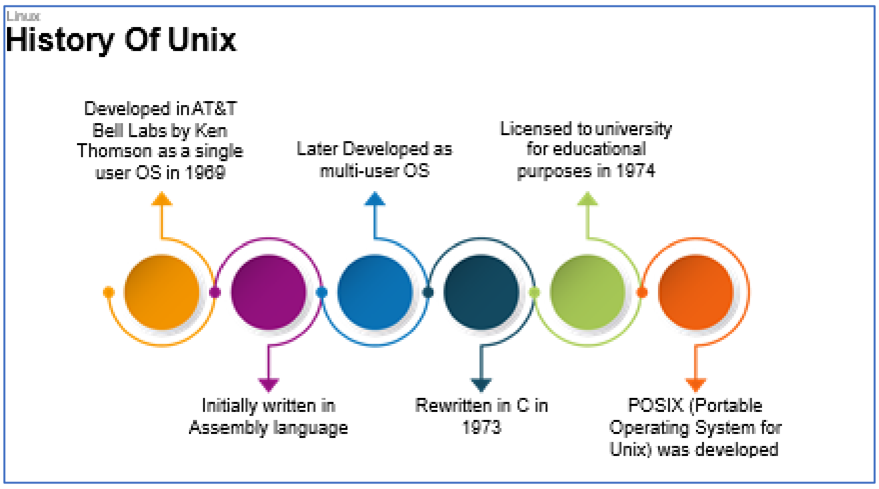
Let us take a quick look at the topics that we will cover in this module:

1. Introduction to Linux
2. Linux Distributions
3. Package Management Tools & Repository
4. Command Line Basics
5. Installing Linux
6. Installing RPM & YUM

Facilitator Notes:

Inform the participants about the topics that they will be learning in this module.

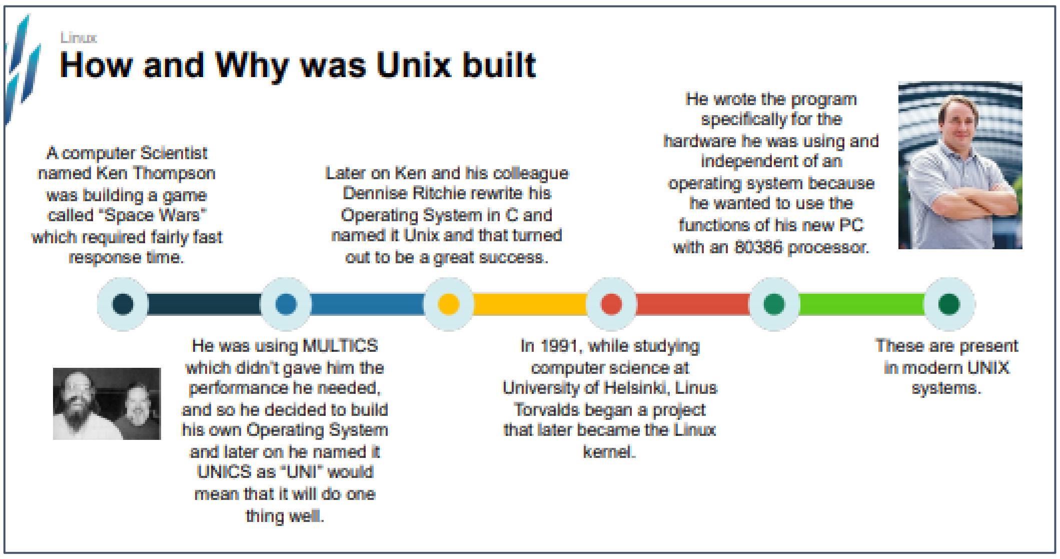
5.1 History of UNIX


Linux
History Of Unix
Developed in AT&T Bell Labs by Ken Thomson as a single user OS in 1969
Initially written in Assembly language
Later Developed as multi-user OS
Rewritten in C in 1973
Licensed to university for educational purposes in 1974
POSIX (Portable Operating System for Unix) was developed

The UNIX system developed by researchers who needed a set of modern computing tools for help in projects. The system allows a group of people working on a project to share selected data and programs keeping other information private.

Universities and colleges play a major role in the popularity of the UNIX operating system by the "four-year effect." The UNIX operating system became available in 1975, Bell Labs offered it to educational institutions at a nominal cost. The schools used it in computer science programs, ensuring that students get familiar with it. UNIX is an advanced development system. The students became acclimated to a sophisticated programming environment. As students worked the way up the ladder in the commercial world, the UNIX operating system found its way into the industry.

Computer Systems Research Group (CSRG) at the University of California at Berkeley made significant additions and changes to the UNIX operating system. One version of the UNIX system is the Berkeley Software Distribution (BSD) (or Berkeley UNIX). The other version is UNIX System V (SVR4), which descended from versions developed and maintained by AT&T and UNIX System Laboratories.



Linux

How and Why was Unix built

A computer Scientist named Ken Thompson was building a game called "Space Wars" which required fairly fast response time.

He was using MULTICS which didn't gave him the performance he needed, and so he decided to build his own Operating System and later on he named it UNICS as "UNI" would mean that it will do one thing well.

Later on Ken and his colleague Dennise Ritchie rewrite his Operating System in C and named it Unix and that turned out to be a great success.

In 1991, while studying computer science at University of Helsinki, Linus Torvalds began a project that later became the Linux kernel.

He wrote the program specifically for the hardware he was using and independent of an operating system because he wanted to use the functions of his new PC with an 80386 processor.

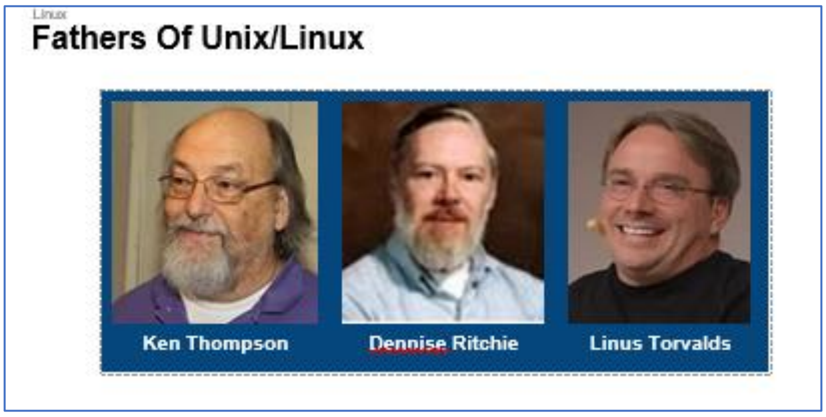
These are present in modern UNIX systems.

A portable operating system is one that can run on many different machines. More than 95 percent of the Linux operating system is written in the C programming language, and C is portable because it is written in a higher-level, machine- independent language. (The C compiler is written in C.) Because Linux is portable, it can be adapted (ported) to different machines and can meet special requirements.

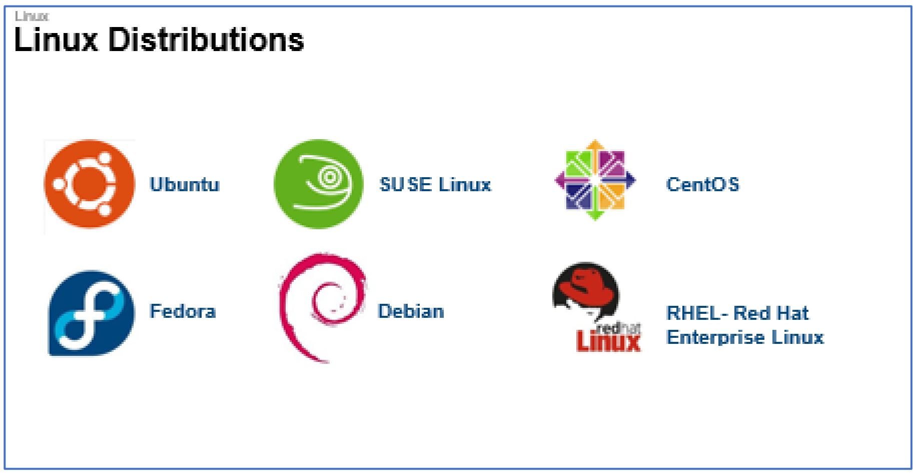
For example, Linux is used in embedded computers, such as the ones found in cellphones, PDAs, and the cable boxes on top of many TVs. The file structure takes full advantage of large, fast hard disks. Equally important, Linux was originally designed as a multiuser operating system it was not modified to serve several users as an afterthought. Sharing the computer's power among many users and giving them the ability to share data and programs are central features of the system.

Because it is adaptable and takes advantage of available hardware, Linux runs on many different microprocessor-based systems as well as mainframes. The popularity of the microprocessor-based hardware drives Linux; these microcomputers are getting faster all the time, at about the same price point. Linux on a fast microcomputer has become good enough to displace workstations on many desktops. Linux benefits both users, who do not like having to learn a new operating system for each vendor's hardware, and system administrators, who like having a consistent software environment.

The advent of a standard operating system has aided the development of the software industry. Now software manufacturers can afford to make one version of a product available on machines from different manufacturers.



Ken Thompson first built UNIX in assembly language. Later, Dennise Ritchie helped him to build Unix in C language.



Following are the key details of Linux:

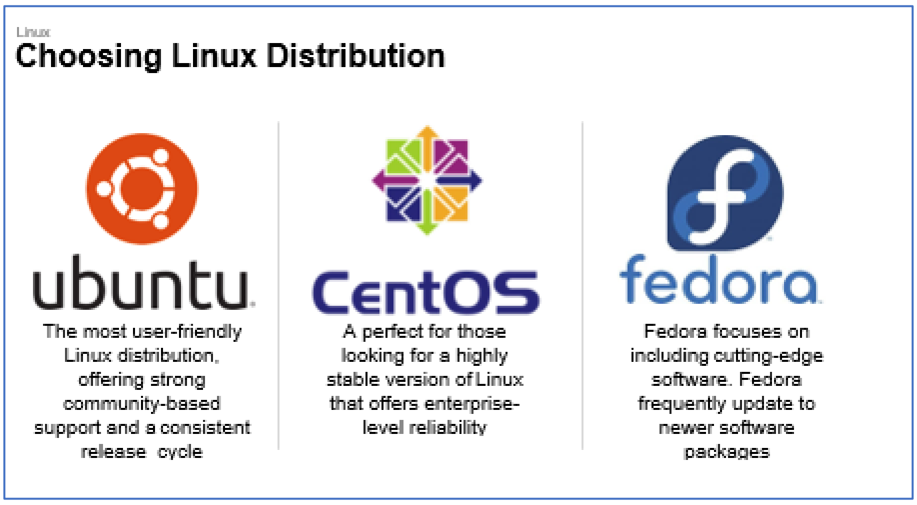
- Linux is an open-source (Freely distributable) version of Unix. It was first developed by Linus Torvalds.
- He began to work on Linux in 1991 when he was a student at the University of Helsinki in Finland.
- Linus still maintains the Linux kernel, which is, the lowest-level core component of the operating system.

Unix Flavors:

- AIX by IBM
- Solaris by Sun Microsystems
- HP-UX by Hewlett-Packard Company
- IRIX by Silicon Graphics, Inc.
- FreeBSD by FreeBSD Group
- GNU / Linux by Open Source Movement
- MacOS by Apple Computer, Inc.
- SCO Unix by The Santa Cruz Operation Inc.

5.2 Choosing Linux Distributions

Linux is considered to be most secure OS amongst windows and Mac OS and is less prone to virus attacks and that is the primary reason it is used in all major Government & Private Domains



Linux

Choosing Linux Distribution

ubuntu
The most user-friendly Linux distribution, offering strong community-based support and a consistent release cycle

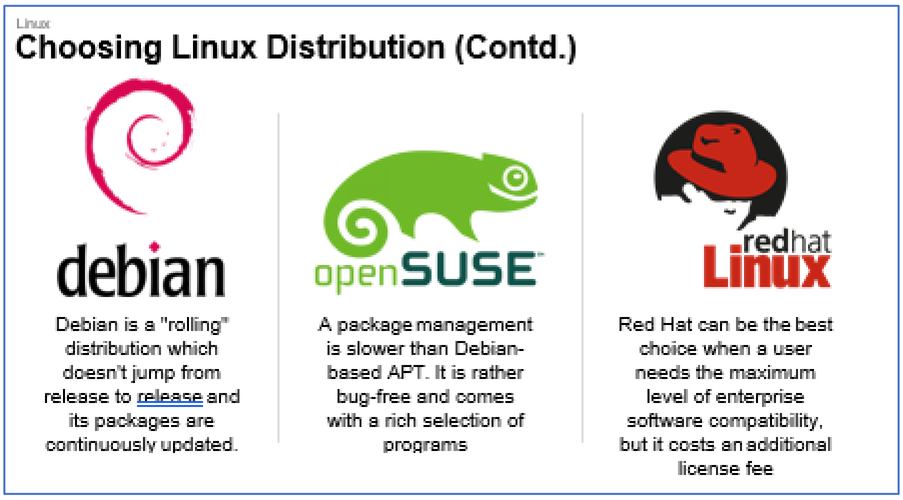
CentOS
A perfect for those looking for a highly stable version of Linux that offers enterprise-level reliability

fedora
Fedora focuses on including cutting-edge software. Fedora frequently update to newer software packages

Ubuntu is the most user-friendly Linux distribution, offering strong community-based support and a consistent release cycle. It is known for its reputation. Ubuntu updates the latest software versions very frequently. The disadvantage of frequent updates is that it's hard to keep bugs from entering into the system. In fact, Ubuntu is not intended to be updated with all the latest software as time goes on. It is designed for the opposite, to be stocked with long-tested software and only upgrading them with critical and security-related fixes. Ubuntu is the best choice for newbies.

CentOS is perfect for those looking for a highly stable version of Linux that offers enterprise-level reliability. The cost of stability is that the software versions included with CentOS are rarely the latest. It comes with the same set of well- tested and stable Linux kernel and software packages that form the basis of its parent, Red Hat Enterprise Linux. Security features include an excellent firewall and SELinux, a policy enforcement mechanism that prevents wayward applications from ever causing security problems.

Fedora focuses on including cutting-edge software. Fedora frequently update to newer software packages. Since Fedora's priorities tend to lean towards enterprise features, rather than server usability; some bleeding edge features occasionally alienate some users. On the other hand, since Fedora is not as popular as Ubuntu and CentOS, it may sometimes be harder to find the app users are looking for. They will be stuck building from source instead of just installing it from the repositories.
Building from source isn't all that hard, but it won't allow users to automatically update that program. Fedora is recommended for advanced Linux administrators


Linux

Choosing Linux Distribution (Contd.)

debian
openSUSE
redhat Linux

Debian is a "rolling" distribution which doesn't jump from release to release and its packages are continuously updated.

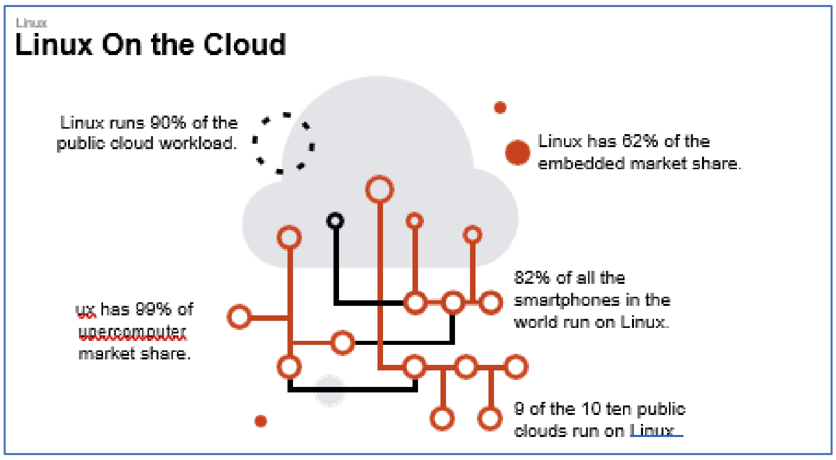
A package management is slower than Debian-based APT. It is rather bug-free and comes with a rich selection of programs

Red Hat can be the best choice when a user needs the maximum level of enterprise software compatibility, but it costs an additional license fee

Debian is a "rolling" distribution which doesn't jump from release to release and its packages are continuously updated. With a moving base of packages, each new change can potentially introduce some problems. Some users consider Debian as one of the stable distributions, as users are quick to report broken features and developers are quick to fix them. Overall, it provides an "unstable" repository.
Debian is a usable and responsive distribution, but it isn’t one we’d recommend for complete beginners.

openSUSE package management is slower than Debian-based APT. It is rather bug- free and comes with a rich selection of programs. openSUSE is a little more difficult to setup by newbies.

Red Hat can be the best choice when a user needs the maximum level of enterprise software compatibility, but it costs an additional license fee. Red Hat Enterprise Linux (RHEL) is for enterprise-level servers. RHEL requires an extra license fee to Red Hat to access their non-free software components. RHEL is stable and handles heavy loads well. The main reason to use RHEL would be if a user is running a software that has RHEL in its list of supported operating systems. This means it aims at larger businesses. If a user is not running software that requires RHEL but want to take advantage of its reliability they can choose Ubuntu or CentOS instead

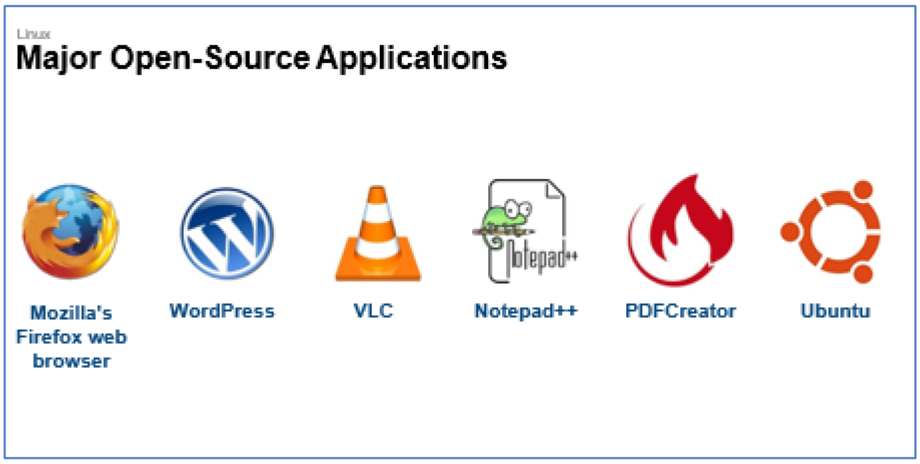

Linux
Linux On the Cloud
Linux runs 90% of the public cloud workload.
Linux has 62% of the embedded market share.
82% of all the smartphones in the world run on Linux.
ux has 99% of upercomputer market share.
9 of the 10 ten public clouds run on Linux.

OpenStack is a set of software tools for building and managing cloud computing platforms for public and private clouds. Backed by big companies in software development, it is the future of cloud computing.

OpenStack is a cloud operating system controlling a large pool of computing, storage, and network resources in a data center. It gets managed and provisioned through APIs with common authentication mechanisms

"**Open source**" refers to something people can change and share because its design is accessible. The term originated in the context of software development to name a specific approach to creating computer programs. "Open source" designates a broader set of values that we call "the open source way." Open-source projects, products, or initiatives embrace principles of open exchange, collaborative participation, rapid prototyping, transparency, meritocracy, and community-oriented development.

What is an Open-Source Software?
An Open source is a software with source code that anyone can inspect, change, and enhance. "Source code" is the part of the software that computer users don't ever see. It is the code programmers can manipulate to change how a piece of software - a "program" or "application - works.



A few other Popular Open-Source applications are:

- Python programming language
- Apache
- HTTP web server
- PHP scripting language

5.3 Command Line Basics

5.3.1 Basic Shell & Variables

Following are the key details of UNIX Shell

- Unix shell is a command language interpreter.
- Its primary purpose is to translate command lines typed at a terminal into system actions.
- The shell is a program through which other programs get invoked.
- There are several different Unix shells like the C shell (csh), the Bourne shell and the Korn shell.
- It is a user program environment provided for user interaction.
- It executes commands read from the standard input device (keyboard) or a file.

In UNIX there are two types of shells:

- The Bourne shell (includes sh, ksh, and bash)
- The C shell (includes csh and tcsh)

The Bourne-type shells follow:

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

The C-type shells follow:

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

Following are the key details of Variables

PATH Variable

- The search path for command / file is stored in an environment variable called PATH
- A typical PATH setting might look something like this:

$echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin

- The path is searched in order, so if there are two commands with the same name, the one that is found first in the path will be executed
- You can add new directories to your search path on the fly, but the path is usually set in shell setup files

SHELL Variable: Shell Variable tells the type of shell user is currently in.

$echo $SHELL: /bin/bash which you are

PATH is an environmental variable in Linux and other Unix-like operating systems. It tells the shell which directories to search for executable files (ready-to-run programs) in response to commands issued by a user. It increases the convenience and safety of operating systems. It the single most important environmental variable.

5.3.2 Type of Commands

Internal Commands	External Commands
"Internal Commands are built into theshell. Example: **cd command** is built-in command"	"External commands are external Programs stored in the file. Example: **ls**"
"Shell doesn't start a separate process to start internal commands"	External commands need the shell to fork and exec a new sub-process; this takes some time; especially on a busy system

5.3.3 Categories Of Commands

1. **General Purpose Commands** - commands used in day to day working on Linux Terminal.
2. **Communication Commands**- commands used to communicate to other Terminal
3. **File and Directory Handling Commands**- commands used to handle Files and Directories.

5.3.3.1 General Purpose Commands

a. **$ pwd –** Shows the present working directory
b. **$ who –** Shows who is logged on
c. **$ whoamI –** Shows the login name of the current user

Examples:

pwd

```
root@ubuntu:~# pwd
/root
```

who

```
$ who
dubai :0 2019-11-26 10:39 (:0)
```

whoami

```
$ whoami
Dubai
```

d. **“date”** command prints or sets the system date and time
e. **date [-u] [datestr]**
f. **$ date** prints date and time of the server
g. **$ date –u display** (or set) the date in Greenwich Mean Time (GMTuniversal time)

```
$ date '+DATE: %m/%d/%y%nTIME:%H:%M:%S'
```

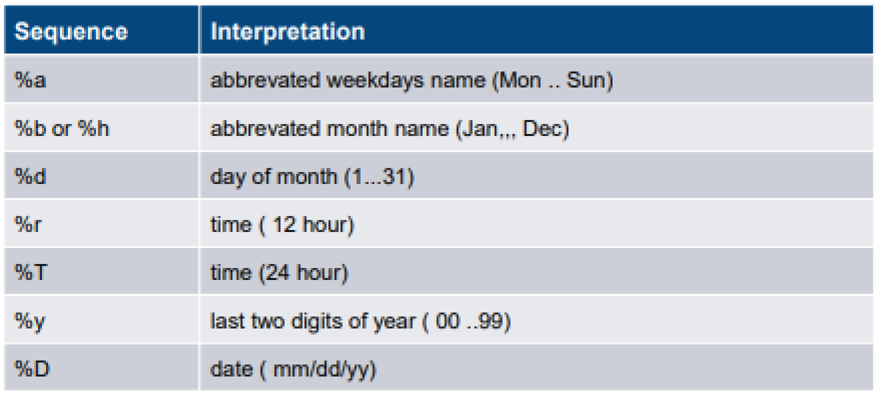
– Lists the time and date in the below format:

```
DATE: 11/30/19
TIME:02:05:02
```

```
$ date -r Dubaimall
```

– Displays the last modification time of the file “Dubaimall”

```
Sun Nov 24 09:54:20 PST 2019
```



Sequence	Interpretation
%a	abbrevated weekdays name (Mon .. Sun)
%b or %h	abbrevated month name (Jan,,, Dec)
%d	day of month (1...31)
%r	time (12 hour)
%T	time (24 hour)
%y	last two digits of year (00 ..99)
%D	date (mm/dd/yy)

date [OPTION]... [+FORMAT]

```
$ date "+Today is %a %m %Y"
Today is Sat 11 2019
```

h. $ which <command_name>
Shows the path of the specified command

i. $ type <command_name>
Shows the type of the specified command
shell built-in or file path if it is external

$which tells which instance of command is being executed when the command is run.The specific instance that is being executed is the one that is first in the PATH.But, there are some instances where $which will return something other than a pathname. If commands are aliased or if the command is a built-in shell command, which may report that as well.

Example:

```
linux2 [41]# which gcc
 /usr/local/bin/gcc
linux2   [42]#   which
mem
 mem: aliased to quota
–v  linux2  [43]#  which
time
 time: shell built-in command
$ type which
which is /usr/bin/which
$  type  date
date          is
/bin/date
$ type ls
ls is a tracked alias for /bin/ls
```

j. $ file list1
Shows the type of the file whether regular file or directory file or some other file

k. $ cal
Shows the calendar of the current month / year

l. $ bc : 2.3+5.4
Performs the mathematical calculations involving integers as well as floats bc is a filter command

```
$ file DigitalE1.txt
DigitalE1.txt: ASCII text
drwxrwxr-x 2 dubai dubai 4096 Nov 30 03:42 dubaimarina
lrwxrwxrwx 1 dubai dubai 9 Nov 30 03:43 dubaim -> dubaimall
$ file dubaimarina
dubaimarina: directory
```

```
$ file dubaim
dubaim: broken symbolic link to dubaimall
$ cal
 November 2019
Su Mo Tu We Th Fr Sa
 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

$ bc
bc 1.07.1
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006, 2008, 2012-2017 Free
Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
3+2+4+6
15
```

m. Echo command can be used to

Display a message
Evaluate shell variables

Options

-n do not output the trailing newline
-e enable interpretation of the backslash-escaped characters
-E disable interpretation of backslash – escaped characters

Examples

```
– $ echo unix operating system
– $echo –n “unix operating system”
– $ echo –e “Unix\n is an OS\n OS\t Operating System”
```

The argument to the echo command can be used without quotes, with double quotes, with single quotes.

Single quotes(‘’): Turns off the meaning of all metacharacters enclosed within them except the single quote itself
Double quotes(“”): Allow variable evaluation and command substitution

Example

```
– $ echo ‘ $55 amounts to how much in INR? ’
– $ echo “ $55 amounts to how much in INR? ”
```

Evaluate shell variables: echo command interpolates the value of variables

– echo $HOME

HOME is an environmental variable, to interpolate its value the $ symbol is used in echo command

– echo $PATH

Prints the PATH environmental variable value

Command Substitution: Back tick quotes are used with echo command to execute the command enclosed within

$echo “Date is: `date` , the default format”
$echo –e “The path you set is:$PATH\n”

n. **tty -** prints the file name of the terminal connected to standard input
$ tty /dev/pts/0

5.3.3.2 File Handling Commands

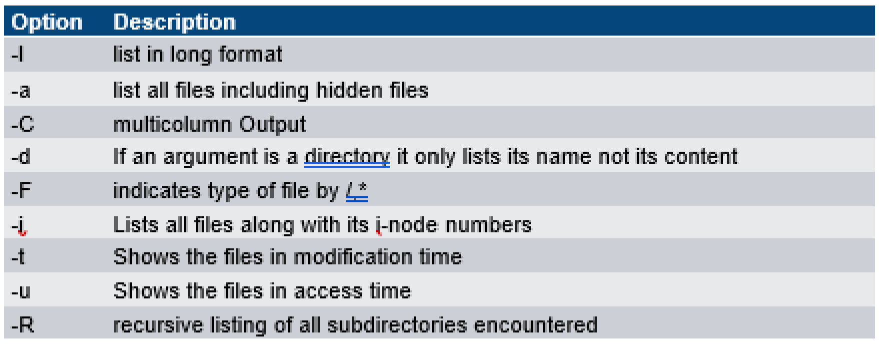
Path names can be absolute or relative.

Absolute Path: - As the name suggests absolute path is the complete path of a file or directory from the “/” root file system. It means complete path of the file/directory starting from root.

Absolute path always starts with “/”. For example:
/var/log/messages
Relative Path: - Relative path of a file is path of file in relation to present working directory.

For example : If the present working directory is /var/log/messages user can change directory and move into var folder by “cd ../../” command

a. **$ls** command lists files & Directories



Option	Description
-l	list in long format
-a	list all files including hidden files
-C	multicolumn Output
-d	If an argument is a directory it only lists its name not its content
-F	indicates type of file by /.*
-i	Lists all files along with its i-node numbers
-t	Shows the files in modification time
-u	Shows the files in access time
-R	recursive listing of all subdirectories encountered

Wild Card Characters

– ***** : Zero or more characters

ls m* - lists all the files starting with m

- **? :** Single Character
  ls m? - lists all the files starting with m and having one character after that

- **[] :** Any character from all the characters within []
  ls [aeiou]* - lists all the files starting with a or e or i or o or u

- **- :** Specifies range
  ls [1-9]* - list everything which has name starting from 1 to 9.

- **! :** Works as not operator
  ls[!x-z]* - lists all the files not starting with any character from the range x-z

b. $ cat
To concatenate files
To display contents of one or more ordinary files
To create an ordinary file

c. $ touch: Change file access and modification time.
d. mkdir [-p] dirname

- Makes a directory of a given dirname
- The dirname can contain the full path prefix of the directory to be created
- More than one directories can be created at the same time
- When executed with –p option, it does not give any error if the directory dirname already exists
- When executed with option –p, it makes parent directories in the path, if needed (if any parent directory in the path is not available)

e. cd [directory]

- Changes working directory to the directory, if specified; otherwise to the home directory
- cd .. moves to the parent directory and cd. keeps you in the current directory
- cd - moves you to the last accessed directory.

f. rmdir command is used to delete only empty directories
$rmdir emptyDir
rm command is used for deleting unwanted files / directories
$ rm [-i] file …
It is interactive removal (option –i) of specified files
$ rm -r directory …
It is recursive deletion of all the files within the specified directories and also the directories themselves

g. cp command is used to delete only empty directories
$ cp –i file1 file2
Copies file1 to file2
-i - informs user before overwriting, if file2 exists

$ cp file1 file2 … dest_directory
Copies multiple files in the specified existing directory
$ cp -r directory1 directory2 … dest_directory
Recursively copies files from directory1, directory2 etc. to the dest_directory
Note: Shell Meta-characters can also be used with "cp"

h. **mv** command changes name of the file or moves the file to the specified destination path
$ mv file1 new-file
Renames file1 as new-file
$ mv file1 file2 … dest_directory
Moves multiple files to the specified existing directory
$ mv directory1 directory2 … dest_directory
Moves one or more directory subtrees to an existing or new dest_directory

i. **umask** is a Unix environment variable which automatically sets file permissions on newly created files.
The value of argument can be calculated by subtracting the mode as default from the current default mode.
E.g. Current default mode is 0666 and user wants it as 0644 then 666 –644 = 022 will be the parameter which needs to pass with "umask" command

5.4.3.3 File Comparison Commands

- **cmp:** cmp command compares two files and reports the location of first mismatch
- **comm:** comm compares two sorted files and produces result in three columns
- **diff:** diff command compares two files and proposes the changes in first file to make two

5.3.4 Networking Commands

With the increasing number of most Linux-based network operating systems and Linux-based projects like OpenStack, Linux skills have become a necessary condition for networking professionals. ... First, it has become increasingly obvious that most network operating systems are based on some variant of Linux.

Command	Use-Case
Ping	Very helpful command to test the connectivity between the systems. It works on ICMP protocol. It is an TCP/IP command to troubleshoot the connectivity & reachability. It is also used for name resolution for IP Address. Syntax: ping <ip/dns name>
Nslookup	It is majorly used to retrieve information for DNS names by interactively querying the name servers. It provides domain names &

	ip addresses. Syntax: nslookup <options> <dns name>
Netstat	It is highly used to print the information of the ports used by the system based on the protocols, it also provides the information for the interface statistics, routing tables, network connections & multicast memberships. Syntax: netstat <option 1> <option 2> …
Ifconfig	It was used very widely earlier to retrieve the ip addresses of all of the available network interfaces in the system. It is still available on all of the linux systems. In windows, ipconfig is used. Now, instead of this command, “ip” command is been used that is discussed next. Syntax: ifconfig <network interface name> Or Only run “ifconfig” to get the information for all of the network devices.
Ip	It is used to assign ip & troubleshoot ip addresses attached to a system. In addition to that, it can also display information for the network interfaces, it can manipulate routing as well. Syntax: ip <options> The complete command is build using the options for a particular use-case.
Route	This command is used to get all the information regarding the routing tables / ip tables. Syntax: route
Dig	It is the most powerful command to get the dns information as it provides all ip’s associated with a dns. It also provides cname. Syntax: dig <dns name>
Traceroute	It is also a very powerful command as it can trace the packets travelling through the network with the number of hops travelled to reach the destination.

	Syntax: traceroute <ip address>

5.3.5 Using Command Line to get Help

"man <command>" *in Linux is used to display user manual of any command that can be run in terminal It provides a detailed view of the command which includes:*

- *NAME,*
- *SYNOPSIS,*
- *DESCRIPTION,*
- *OPTIONS,*
- *EXIT STATUS,*
- *RETURN VALUES,*
- *ERRORS,*
- *FILES,*
- *VERSIONS,*
- *EXAMPLES,*
- *AUTHORS*
- *SEE ALSO*

man command formats and displays on-line manual pages.

- *The manual pages are divided into a logical group of commands called the sections. Sections are numbered from 1 through 9.*
- *Example: Commands are 1, system calls are 2, library function is 3, file formats are 5, & management commands are 8*
- *If specified in the section , man command looks only in the section of the manual.*
- *The command man 2 open displays the manual for the system call open.*

Info: *Info command reads documentation in the info format. It gives detailed information for a command when compared with the man page. The pages are made using the texinfo tools which it can link with other pages, create menus and easy navigation.*
Syntax: info [OPTION]... [MENU-ITEM...]

Various Options available with info command are as below:

- ***-a, –all:*** *It uses all matching manuals.*
- ***-k, –apropos=STRING:*** *It looks up STRING in all indices of all manuals.*
- ***-d, –directory=DIR:*** *It adds DIR to INFOPATH.*
- ***-f, –file=MANUAL:*** *It specifies Info manual to visit.*
- ***-h, –help:*** *It displays this help and exit.*
- ***-n, –node=NODENAME:*** *It specifies nodes in first visited Info file.*
- ***-o, –output=FILE:*** *Its output is selected nodes to FILE.*
- ***-O, –show-options, –usage:*** *It goes to command-line options node.*
- ***-v, –variable VAR=VALUE:*** *It assigns VALUE to Info variable VAR.*
- ***–version:*** *It displays version information and exit.*
- ***-w, –where, –location:*** *It prints physical location of Info file*

5.4 DevOps and System Administrator

Devops and sysadmins: They have many common tasks but there are also vital differences that we should always be aware of.

Devops vs. system administrator: what is the difference?

They are similar, but different. Do you remember the last time a system administrator came up with the claim that he can do everything a devops guy can do? If you do, then welcome to the part where almost everyone is confused and none is.

In this section, we will go forward and discuss the differences between a devops and system administrator. We will try to explain it from the historical viewpoint and also try to understand them as per their job roles. The job market for devops and system administrators is drastically changing. Current generation of engineers know this and are always improving their skills like the ones from LiveEdu that share their knowledge online with peers and work on projects collaboratively.

However, before we get started, it is important to understand each of the terms. What is "devops"?

According to Wikipedia, devops is defined as below.
"Devops is a software development and delivery process that emphasizes communication and collaboration between product management, software development, and operations professionals. It seeks to automate the process of software integration, testing, deployment and infrastructure changes by establishing a culture and environment where building, testing, and releasing software can happen rapidly, frequently, and more."

What is a "system administrator"?

"A system administrator, or sysadmin, is a person who is responsible for the upkeep, configuration, and reliable operation of computer systems; especially multi-user computers, such as servers."

A clear difference

As you can see, there is a clear difference between the two. However, there is still too much confusion among IT specialists and HR managers. The term "devops" has been abused multiple times by companies to get things done. There are also many companies and startups that think devops guys can do everything and will solve their problems once hired. Devops professionals should not be responsible for clearing all the mess that a company have made up to that point.

History behind the Devops term misuse

- A decade ago, there were separate roles such as testers, sysadmins, developers, database administrators, and so on. It worked at that time as no one complained. However, as time passed, the market changed and cloud computing arrived.

- With the inception of the cloud, a lot of things started to get automated, especially the ones that were done by a sysadmin. It also took away most of the work done by network engineers and database administrator.

- With most of the work for the sysadmins becoming automated, they were asked to support the developers to become more effective and also help the end users. This type of work slowly gave birth to "devops." However, devops' tasks are not limited to what sysadmins do.

- You can read more about the history from MightBigMinus on Reddit where he explains every single bit of change that happened to the profession. It is a simple perspective offered by the Reddit user which makes sense.

- In the end, devops' aim is to make every section of the IT company collaborative in nature. According to Jeff Knup, devops is meant to denote a close collaboration and cross-pollination between what were previously purely development roles, purely operations roles, and purely QA roles.

Devops vs. system administration

Until now, it should be clear that the two roles are different in nature. There are many tasks that are common to both devops and a sysadmin, however, there are also vital differences that we should always be aware of. For example:

1. Devops' job is to collaborate on a high-level and ensure synergy in each section of the company. A sysadmin guy is more focused on configuring, keeping up and maintaining servers and computer systems.

2. Devops guys are known for their experience working on a product from end-to-end while sysadmins are only confined to smaller scope and responsibility.

3. Devops guys can do everything a sysadmin does, but a sysadmin cannot do everything a devops guy does.

Source: *An article by Damian Wolf*

5.5 Linux Installation

For installing Official Ubuntu follow the link: https://ubuntu.com/download/desktop

How to Install Linux (Alternate Options will be taken with Advanced Linux in forthcoming Semesters)

5.5.1 Choose a boot option

This is a proceed-at-your-own-risk tutorial. Also, note that I'll be referring to the old

PC as the
“destination system.”
Here’s the overall process in a nutshell:

- Step one: Download a Linux OS. (Recommended doing this, and all subsequent steps, on your current PC, not the destination system. Although the latter is an option if it’s malware- free and in decent working order, everything will get done faster and more easily on your primary machine.)

- Step two: Create a bootable CD/DVD or USB flash drive.

- Step three: Boot that media on the destination system, then make a few decisions regarding the installation.

The first part is easy: Just download Linux from Mint or Ubuntu or whatever site hosts the version you want. That download will likely consist of a single ISO file. Note: An older computer may have a 32-bit processor, which won’t work with 64-bit versions of Linux.

The second part -- creating boot media -- requires a little thought. The fastest, easiest method is to use a flash drive, even if the destination system has a CD/DVD drive. Indeed, the only reason not to go the flash-drive route is if the destination system won’t boot (or boot properly) from one. (I’ve encountered this problem a few times, even after tweaking the BIOS boot settings and actually selecting “USB drive” from a pop-up boot menu.)

Advise: Try a flash drive first. If it doesn’t work, you can always use that same ISO file to create a bootable CD later.

How big a drive does you need? It depends on the size of the Linux distro. The latest versions of Mint and Ubuntu run about 1.8GB and 1.5GB, respectively, so a 2GB drive should suffice. Make sure it doesn’t contain any important data, as it’ll need to get wiped as part of this procedure.

5.5.2 Build your boot drive

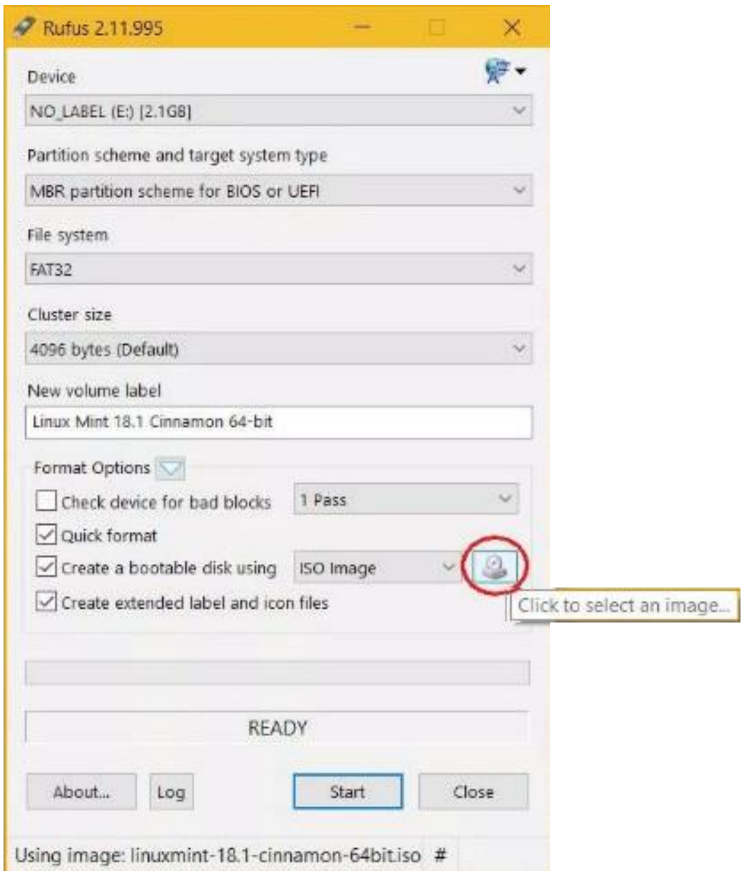
Once you’ve downloaded your Linux ISO, you’ll need a utility that can create a bootable flash drive. I’m partial to Rufus, which is fast, free and easy to use. Download the portable version; there’s no need to actually install it, because most likely you’ll just run it once.

Step 1: Plug in your flash drive (ignoring or closing any prompts that pop up), then run the Rufus utility.
Step 2: In the Device field, at the very top, make sure your flash drive is the one selected. If not, click the pull-down and select it.
Step 3: Near the checkbox marked “Create a bootable disk using,” click

the little disk icon and navigate to the Linux ISO file you downloaded. It's most likely in your Downloads folder. Click it, then click Open.

Step 4: If you like, you can change the "New volume label" field to something like "Linux,"

but it's not necessary. Click Start, then wait while the drive is formatted and the ISO installed.



Rufus 2.11.995

Device: NO_LABEL (E:) [2.1GB]

Partition scheme and target system type: MBR partition scheme for BIOS or UEFI

File system: FAT32

Cluster size: 4096 bytes (Default)

New volume label: Linux Mint 18.1 Cinnamon 64-bit

Format Options

- [ ] Check device for bad blocks — 1 Pass
- [x] Quick format
- [x] Create a bootable disk using — ISO Image
- [x] Create extended label and icon files

Click to select an image...

READY

About... Log Start Close

Using image: linuxmint-18.1-cinnamon-64bit.iso #

5.5.3 Get Ready to Boot

Now it's time to turn your attention to the destination system. It doesn't matter what condition it's in or even if it's riddled with malware; you just need it to be able to boot from a flash drive. That may mean venturing into the BIOS and changing the boot order, which by default almost certainly puts the hard drive first. Some systems do offer a pop-up boot menu that lets you choose what device to boot from without having to monkey with the settings. If yours does, count yourself lucky.

For example, I did some testing with a years-old HP Pavilion dm1z. When you first power it on, there's no boot menu -- just a blank screen and then the Windows startup screen. So I did a quick web search for "Pavilion dm1z boot menu" and learned that I need to press either F1 or F10 immediately after powering on the machine. (Turns out it was F10.)

That’s pretty common, though based on past experience, it might also be F2, F9, F12 or even the Delete key. Depends on the system.

Once you’ve found your way into the BIOS, find the boot or startup menu and make sure “flash drive” or “USB drive” is first in the boot order. Then save and exit (usually by pressing F10, but, again, this varies).

5.5.4 One OS or two?

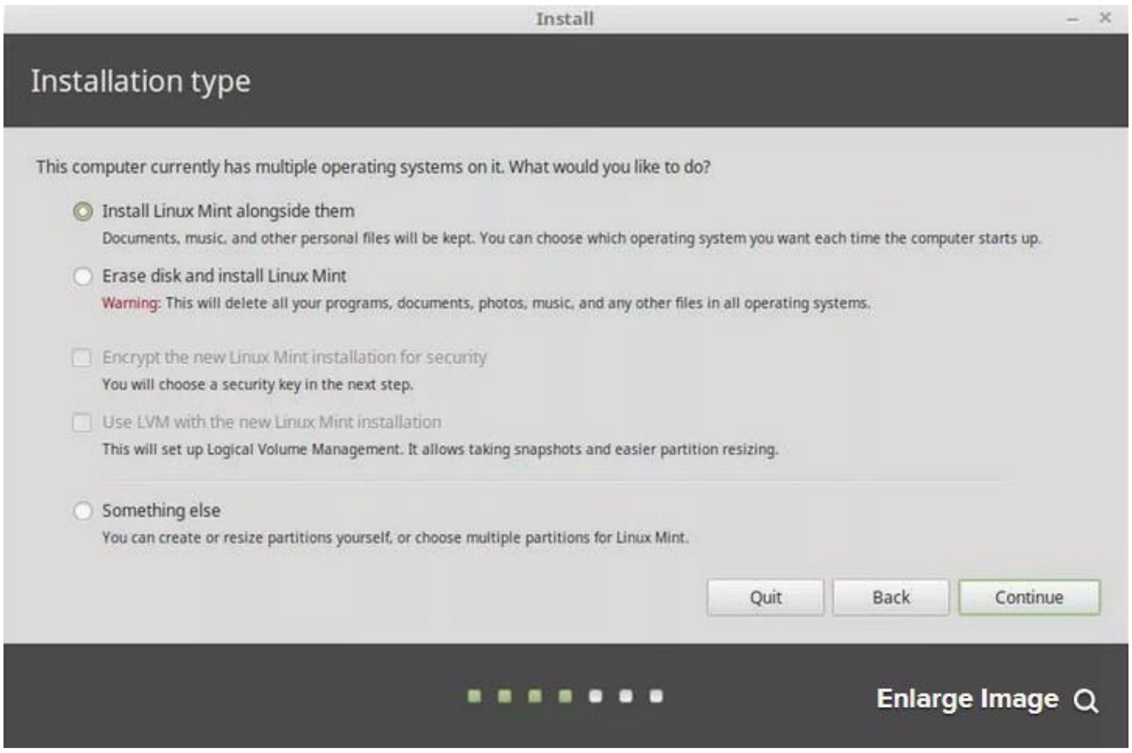
As you probably know already, Linux can boot and run right from the flash drive -- no actual installation required. That’s a great way to test-drive a distro, but this tutorial is about installing the OS, so let’s focus on that option.

The big decision you’ll need to make is whether you want to install Linux alongside your existing OS, which would result in a dual-boot setup, or reformat the hard drive and run only Linux. The former is worth considering if the system has a large drive and can easily accommodate both operating systems, or you still have need for Windows.

This following instructions may vary a bit from one distro to the next, but they’re based on my installation of Linux Mint.

Step one: Boot from the flash drive directly into Linux.

Step two: Double-click the Install Linux icon on the desktop.
Step three: Make any requested selections regarding language, installation of third-party software and so on. Then choose your OS installation preference: alongside the existing OS or erase-and-install.


Install

Installation type

This computer currently has multiple operating systems on it. What would you like to do?

[✓ Install Linux Mint alongside them]
Documents, music, and other personal files will be kept. You can choose which operating system you want each time the computer starts up.

Erase disk and install Linux Mint
Warning: This will delete all your programs, documents, photos, music, and any other files in all operating systems.

Encrypt the new Linux Mint installation for security
You will choose a security key in the next step.

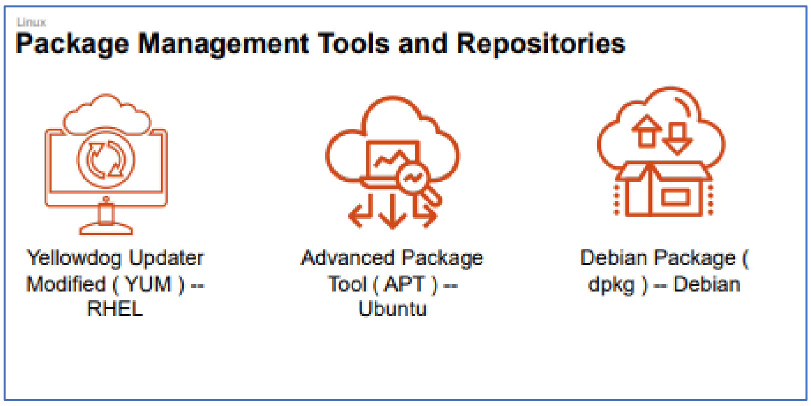
Use LVM with the new Linux Mint installation
This will set up Logical Volume Management. It allows taking snapshots and easier partition resizing.

Something else
You can create or resize partitions yourself, or choose multiple partitions for Linux Mint.

Source: *An article by Rick Broida* *https://www.cnet.com/how-to/how-to-install-linux/*

Reference book: Running Linux - 5th Edition - https://the-eye.eu/public/Books/IT%20 Various/running_linux_5th_edition.pdf
Chapter 2 - Pre Installation and Installation

5.6 Package Management System


Linux
Package Management Tools and Repositories
Yellowdog Updater Modified (YUM) -- RHEL
Advanced Package Tool (APT) -- Ubuntu
Debian Package (dpkg) -- Debian

Package management tools are specific to distribution of Linux being used. Package management is installing and maintaining (updating and removing) software on the system.To fulfill library dependencies, Linux uses Package Management tools like **YUM, APT and dpkg.**

A Repository is a storage location where system fetches and Installs OS updates and applications. A repository is a collection of software hosted on a remote server. It is used for installing and updating software packages on Linux systems.

RedHat Enterprise Linux binaries are with extension RPM which means Red Hat Package manager.

In this section, we learn to install the RPM and YUM tools to manage the packages on your Linux system. Learn to Install, reinstall, upgrade, and remove packages using RPM and YUM.

Introducing Package Management

In the past, many Linux programs were distributed as source code, which a user would build into the required program or set of programs, along with the required man pages, configuration files, and so on. Nowadays, most Linux distributors use prebuilt programs or sets of programs called *packages*, which ship ready for installation on that distribution. In this tutorial, you will learn about *package management tools* that help you install, update, and remove packages. This tutorial focuses on the **Red Hat Package Manager (RPM)**, which was developed by Red Hat, as well as the **Yellowdog Updater Modified (YUM)**, which was originally developed to manage Red Hat Linux systems at Duke University's Physics department.

From a user perspective, the basic package management function is provided by commands. As Linux developers have striven to make Linux easier to use, the basic tools have been supplemented by other tools, including GUI tools, which hide some of the complexities of the basic tools from the end user. In this tutorial and in the tutorial on Debian package management, we focus on the basic tools, although we mention some of the other tools so you can pursue them further.

Prerequisites

To get the most from the tutorials in this series, you should have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this tutorial.

Sometimes different versions of a program will format output differently, so your results may not always look exactly like the listings and figures shown here. In particular, much of the output we show is highly dependent on the packages that are already installed on our systems. Your own output may be quite different, although you should be able to recognize the important commonalities. The examples in this tutorial use a Fedora 20 system unless otherwise noted.

Package Managers

RPM, YUM, and APT (for Debian systems) have many similarities. All can install and remove packages. Information about installed packages is kept in a database. All have basic command-line functionality, while additional tools can provide more user-friendly interfaces. All can retrieve packages from the Internet.

When you install a Linux system, you typically install a large selection of packages. The set may be customized to the intended use of the system, such as a server, desktop, or developer workstation. And at some time you will probably need to install new packages for added functionality, update the packages you have, or even remove packages that you no longer need or that have been made obsolete by newer packages. Let's look at how you do these tasks, and at some of the related challenges such as finding which package might contain a particular command.

RPM

Red Hat introduced RPM in 1995. RPM is now the package management system used for packaging in the Linux Standard Base (LSB). The rpm command options are grouped into three subgroups for:

- Querying and verifying packages
- Installing, upgrading, and removing packages
- Performing miscellaneous functions

We should also note that rpm is the command name for the main command used with RPM, while *.rpm* is the extension used for RPM files. So "an rpm" or "the xxx rpm" will generally refer to an RPM file, while rpm will usually refer to the command.

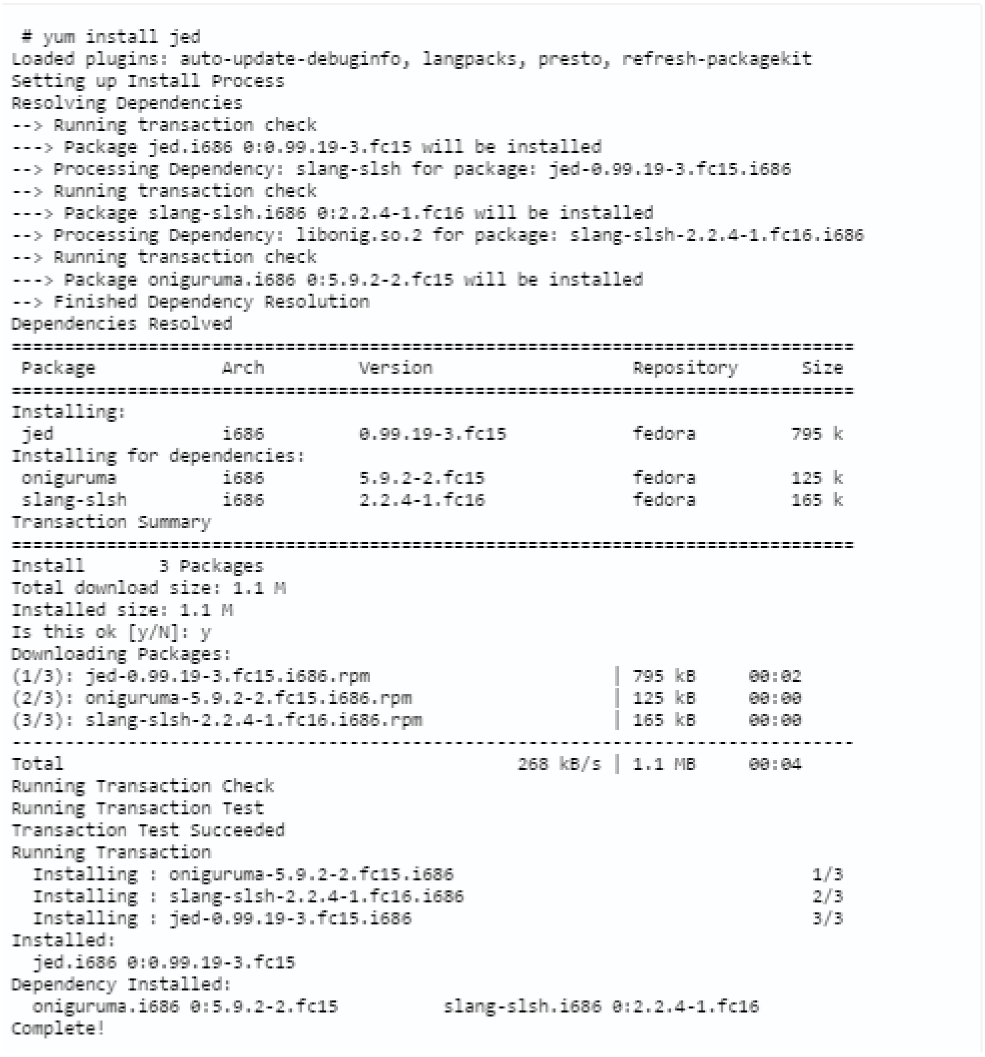
YUM

YUM adds automatic updates and package management, including dependency management, to RPM systems. In addition to understanding the installed packages on a system, YUM is like the Debian Advanced Packaging Tool (APT) in that it works with *repositories*, which are collections of packages and are typically accessible over a network connection.

The dark old days of RPM dependency hell. Well, now, thanks to yum, those days are a thing of the past and software installation became simpler. Just like theother package managers, yum supports basic functions like installing or removing software, plus a bunch of other useful options.

Installing Software

The basic commands for software management are more or less the same as the ones we talked about above, in the Debian part. So, if, for example, you want to install jed, the text editor, just do



```
 # yum install jed
Loaded plugins: auto-update-debuginfo, langpacks, presto, refresh-packagekit
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package jed.i686 0:0.99.19-3.fc15 will be installed
--> Processing Dependency: slang-slsh for package: jed-0.99.19-3.fc15.i686
--> Running transaction check
---> Package slang-slsh.i686 0:2.2.4-1.fc16 will be installed
--> Processing Dependency: libonig.so.2 for package: slang-slsh-2.2.4-1.fc16.i686
--> Running transaction check
---> Package oniguruma.i686 0:5.9.2-2.fc15 will be installed
--> Finished Dependency Resolution
Dependencies Resolved
================================================================================
 Package            Arch         Version                   Repository      Size
================================================================================
Installing:
 jed                i686         0.99.19-3.fc15            fedora         795 k
Installing for dependencies:
 oniguruma          i686         5.9.2-2.fc15              fedora         125 k
 slang-slsh         i686         2.2.4-1.fc16              fedora         165 k
Transaction Summary
================================================================================
Install       3 Packages
Total download size: 1.1 M
Installed size: 1.1 M
Is this ok [y/N]: y
Downloading Packages:
(1/3): jed-0.99.19-3.fc15.i686.rpm                       | 795 kB     00:02
(2/3): oniguruma-5.9.2-2.fc15.i686.rpm                   | 125 kB     00:00
(3/3): slang-slsh-2.2.4-1.fc16.i686.rpm                  | 165 kB     00:00
--------------------------------------------------------------------------------
Total                                          268 kB/s | 1.1 MB     00:04
Running Transaction Check
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : oniguruma-5.9.2-2.fc15.i686                                  1/3
  Installing : slang-slsh-2.2.4-1.fc16.i686                                 2/3
  Installing : jed-0.99.19-3.fc15.i686                                      3/3
Installed:
  jed.i686 0:0.99.19-3.fc15
Dependency Installed:
  oniguruma.i686 0:5.9.2-2.fc15         slang-slsh.i686 0:2.2.4-1.fc16
Complete!
```

This is how the complete output looks like on my Fedora 16 machine. yum also supports the localinstall command, which is very useful if you have a RPM package downloaded locally and you want to install it. We recommend the use of localinstall versus “rpm -i $package” because the package gets added to the yum database so you have a less chaotic environment.

Updating Software

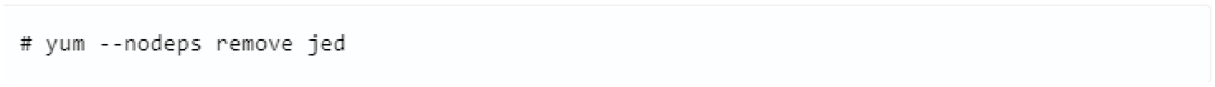
As said before, the commands are pretty similar to some extent, but with yum you don’t have two separate commands like you have with apt*. So a simple “yum update” will update the repository data *and* proceed to the update proper if newer versions of software is found. You can see if there’s something new by using the check-update command, which is close to “aptitude update” as it updates the repo data but doesn’t do anything else.

Searching for Software

Until recently I didn’t like yum’s search command because it yielded too many results, some quite unrelated to what I wanted. It seems that there were others with the same problems, so the developers changed the search function to print only relevant results, and added the “search all” command to emulate the old behavior.

Removing Software

If I want to remove software, of course the command is “yum remove”. This will remove the package and its dependencies. Should you want not to go that route, that is you want to keep the dependencies (we recommend great care here), you should type



```
# yum --nodeps remove jed
```

The yum wiki says this about --nodeps so again, be careful: “--nodeps is only used when a package or system is badly broken. As a general rule if you find you cannot put the screw in the hole with a screw driver you should not go get a hammer.” Of course, you’re supposed to read the yum manual, which explains all the options you can use. Like with text editors, the package manager is a often-used tool, so it’s imperative you know about it so you can be more efficient.

Installing RPM packages

Suppose you want to compile a Fortran program and a colleague tells you to use gfortran command. You might try gfortran –helpgfortran--help, or you might try which gfortranwhichgfortran, or type gfortrantypegfortran. But if your system can't find gfortran, you might see output similar to that shown in Listing 1.

Listing 1. Missing gfortran command

```
[ian@attic-f21 ~]$ gfortran --help
bash: gfortran: command not found [ian@attic-f21 ~]$ gfortran --help bash:
gfortran: command not found...
Install package 'gcc-gfortran' to provide command 'gfortran'? [N/y] n

[ian@attic-f21 ~]$ which gfortran

/usr/bin/which: no gfortran in (/usr/local/bin:/usr/local/sbin:/usr/bin:/
usr/sbin:/bin:/sbin:
/home/ian/.local/bin:/home/ian/bin) [ian@attic-f21 ~]$ type gfortran bash:
type: gfortran: not found
```

If you did not get the helpful suggestion from the second form of output in Listing 1, you might check back with your colleague to find out which package to install. Otherwise, you might just guess that the gfortran command is in the gfortran package. This is often a good guess, but not always the right one and not the right one in this case. We'll see later how to find the right package. Assuming that you know it's really in the gcc-gfortran package and that you downloaded or otherwise acquired a copy of the package, you might try installing it using the rpm command with the -i (for install) option, as shown in Listing 2.

Listing 2. Installing gcc-gfortran with rpm – take 1

```
root@attic-f21 ~rpm -i  gcc-gfortran-4.9.2-6.fc21.x86_64.rpm error: Failed
dependencies:
        libquadmath-devel = 4.9.2-6.fc21 is needed by gcc-gfortran-4.9.2-6.
fc21.x86_64
```

The rpm command knows that the package has a dependency, but unfortunately, it won't help you resolve that dependency. You will need to get the dependent package or packages, try again, and see if there are additional dependencies—and keep doing this until all dependencies are satisfied. One good thing is that you can give the rpmcommand a list of packages to install and it will install them all in the right order if all dependencies are satisfied. So you at least don't have to manually install each piece in the right order.

If you've used Debian's APT, by this time you're probably wishing you had something like the apt-get command, which would simply go and find what you need, including dependencies, and just install it. For RPM-based systems, YUM (or Yellowdog Updater Modified) provides just such a function. Listing 3 shows how to install gcc-gfortran and the required prerequisites using the yum command with the install option. **Note:** Your dependencies may differ according to what you already have installed on your system.

Listing 3. Installing gcc-gfortran using yum

```
root@attic-f21 ~yum install
gcc-gfortran Loaded plugins: langpacks
Resolving Dependencies

 --> Running transaction check

 ---> Package gcc-gfortran.x86_64 0:4.9.2-6.fc21 will be installed

 --> Processing Dependency: libquadmath-devel = 4.9.2-6.fc21 for package:
 gcc-gfortran-4.9.2-6.fc21.x86_64

 --> Running transaction check

 ---> Package libquadmath-devel.x86_64 0:4.9.2-6.fc21 will be installed

 --> Finished Dependency Resolution Dependencies Resolved
 ============================================================
 ==========
  Package                  Arch          Version
                           Repository    Size
 ============================================================
 ==========

 Installing:

  gcc-gfortran             x86_64        4.9.2-6.fc21            updates
                           7.7 M Installing for dependencies:
  libquadmath-devel         x86_64        4.9.2-6.fc21            updates
                            37 k

 Transaction Summary
 ============================================================
 ==========
        Install       1 Package (+1 Dependent package) Total download size: 7.7 M
        Installed size: 18 M Is this ok [y/d/N]: y Downloading packages:

(1/2): libquadmath-devel-4.9.2-6.fc21.x86_64.rpm                    |   37
kB                                                                  00:00

(2/2): gcc-gfortran-4.9.2-6.fc21.x86_64.rpm                         | 7.7
MB                                                                  00:04
-----------------------------------------------------------------------

Total                                                  1.6 MB/s | 7.7
MB                                                     00:04

Running transaction check Running transaction test Transaction test succeeded
Running transaction (shutdown inhibited)

  Installing :libquadmath-devel-4.9.2-6.fc21.x86_64                         1/2

  Installing : gcc-gfortran-4.9.2-6.fc21.x86_64                             2/2

  Verifying    : libquadmath-devel-4.9.2-6.fc21.x86_64                      1/2

  Verifying    : gcc-gfortran-4.9.2-6.fc21.x86_64                           2/2

Installed:

   gcc-gfortran.x86_64 0:4.9.2-6.fc21 Dependency Installed:
   libquadmath-devel.x86_64 0:4.9.2-6.fc21 Complete!
```

The output in Listing 3 shows that YUM has found the x86_64 versions of gcc-gfortran and libquadmath-devel in a repository called “updates” (more on that shortly), and determined the total download size. After you respond “y” to agree to the transaction, it downloaded both packages, and then installed the dependency, followed by gcc-gfortran. You will learn more about dependencies later in this tutorial.

Note: In Listing 3, YUM found the latest version of the gcc-gfortran package which happened to be the same level (4.9.2-6) as the one we attempted to install in Listing 2. You will usually want the latest version of a package, but you can provide additional qualifications if you need an earlier version, or the i686 version instead of the x86_64 version. See the section on specifying package names in the man pages for the yum command.

Package Locations

In the previous section, you learned how to install an RPM package. But where do the packages come from? How does yum know where to download packages from? The starting point is the /etc/yum.repos.d/ directory, which usually contains several *repo*files. This is the default location for repository information, but other locations may be specified in the YUM configuration file, normally /etc/yum.conf. Listing 4 shows the fedora-updates.repo corresponding to the location from which we installed gcc-gfortran on our Fedora 21 system.

A typical repo file is divided into three sections, one for normal packages, one for debug packages, and the last for source packages. Usually, there will be several copies of a distribution's packages available from different locations, or *mirrors*. So the repo file tells yum where to find the latest list of mirrors for each section. Note that the distribution release level and machine architecture are parametrized, so yum would download the list for my x86_64 Fedora 21 system from https://mirrors.fedoraproject.org/metalink?repo=updates-released- f21&arch=x86_64.

In addition to the repository location, the repo file tells whether a particular repository is enabled and whether GPG signatures should be used to check the downloaded packages.

Listing 4. /etc/yum.repos.d/*.repo

```
[ian@attic-f21 ~]$ cat /etc/yum.repos.d/fedora-updates.repo updatesname=Fedora
$releasever - $basearch - Updates failovermethod=priority

#baseurl=http://download.fedoraproject.org/pub/fedora/linux/
updates/$releasever/$basearch/

metalink=https://mirrors.fedoraproject.org/
metalink?repo=updates-released-f$releasever&arch=$

basearch enabled=1
metadata_expire=6h gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-fedora-$releasever-$basearch

skip_if_unavailable=False [updates-debuginfo]
name=Fedora $releasever - $basearch - Updates - Debug failovermethod=priority

#baseurl=http://download.fedoraproject.org/pub/fedora/linux/
updates/$releasever/$basearch/debug/

metalink=https://mirrors.fedoraproject.org/
metalink?repo=updates-released-debug-f$releasever&
```

```
arch=$basearch enabled=0 gpgcheck=1 metadata_expire=6h
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-fedora-$releasever-$basearch

skip_if_unavailable=False

[updates-source]

name=Fedora $releasever - Updates Source failovermethod=priority

#baseurl=http://download.fedoraproject.org/pub/fedora/linux/
updates/$releasever/SRPMS/

metalink=https://mirrors.fedoraproject.org/
metalink?repo=updates-released-source-f$releasever&

arch=$basearch enabled=0 gpgcheck=1 metadata_expire=6h
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-fedora-$releasever-$basearch

skip_if_unavailable=False
```

YUM and RPM use a local database to determine what packages are installed. The metadata about packages that is stored in the local database is retrieved from the enabled repositories. Although you will seldom need to worry about the local database, you use the command yum **cleanyumclean** to clean out various parts of the locally stored information and yum makecacheyummakecache to create the information in your local database for the enabled repos. You might do this if you change your repo configuration, for example.

Removing RPM packages

If you want to remove a package, you can use the remove option of yum, or the -e option of rpm. A test run to remove gcc-gfortran using rpm -erpm-e is shown in Listing 5. If the package can be removed, there is no output.

Listing 5. Test removal of gcc-gfortran root@attic-f21 ~rpm -e --test gcc-gfortran [

Unlike the simulated removal of Debian packages using apt-get, the RPM system does not maintain information on packages that were automatically added, so there is no trivial way to find out which dependencies might also be removed. However, if you specify multiple packages for removal on a single command, then packages without dependencies will be removed before packages that have dependencies.

When you remove packages using rpm, there is no prompt before the packages are removed, unlike when you install packages. However, if you attempt to remove a package that is required for some other package, the operation is not performed and you get an error message as shown in Listing 6.

Listing 6. Removing a dependent package with rpm

```
root@attic-f21 ~rpm -e
libquadmath-devel  error: Failed  dependencies:
                libquadmath-devel        =    4.9.2-6.fc21        is      needed    by
                                         (installed) gcc-gfortran-4.9.2-6.fc21.x86_64
```

If you use yum removeyumremove instead, then you will be prompted after the transaction tests are performed. If the package you are trying to remove is a dependent package for some other installed packages, then YUM will offer to remove those as well as the dependent package, as shown in Listing 7.

Listing 7. Removing a dependent package with yum

```
root@attic-f21 ~yum remove
libquadmath-devel Loaded plugins: langpacks
Resolving Dependencies

 --> Running transaction check

 ---> Package libquadmath-devel.x86_64 0:4.9.2-6.fc21 will be erased

 --> Processing Dependency: libquadmath-devel = 4.9.2-6.fc21 for package:
 gcc-gfortran-4.9.2-6.fc21.x86_64

 --> Running transaction check

 ---> Package gcc-gfortran.x86_64 0:4.9.2-6.fc21 will be erased

 --> Finished Dependency Resolution Dependencies Resolved
 =====================================================================
 ====
  Package                          Arch          Version            Repository
                                   Size
 =====================================================================
 ====

 Removing:

  libquadmath-devel                 x86_64        4.9.2-6.fc21
                                    @updates      18 kRemoving for dependencies:
  gcc-gfortran                     x86_64        4.9.2-6.fc21
                                   @updates      18 M

 Transaction Summary
 =====================================================================
 ====
 Remove          1 Package (+1 Dependent package) Installed size: 18 M
 Is this ok [y/N]: n

 Exiting on user command

 Your transaction was saved, rerun it with:


yum load-transaction /tmp/yum_save_tx.2015-07-27.22-01.amzaZh.yumtx
```

If you know where the RPM files are located, or have downloaded them, you can also update them by using the rpm command. This is similar to installing, except that you use the -U or the -F option instead of the -i option. The difference between these two options is that the -U option will upgrade an existing package **or** install the package if it is not already installed, while the -F option will only upgrade or *freshen* a package that is already installed. Because of this, the -U option is frequently used, particularly when the command line contains a list of RPMs. This way, uninstalled packages are installed, while installed packages are upgraded. Two other options, -v (verbose) and -h (hash marks), are often used to give progress indication. Listing 9 shows how to update the cairo package and its cairo-gobject dependency using the rpm command. We have the cairo rpm already downloaded in root's home directory, while we retrieve the cairo-gobject package from one of the update mirrors.

In a nutshell, we learnt:

1. Linux Evolution & Popular OS
2. Command Line Basics
3. Basic Shell & Variables
4. Using the Command Line to Get Help
5. Networking Commands
6. Linux Installation
7. Introduction to Package Management System: RPM and YUM