LPI Linux Essentials

Course Navigation

- **Open Source** Section 1
- **Finding Your Way** Section 2
- **Command Line** Section 3
- **Operating System** Section 4
- **Security** Section 5
- **Exam Preparation**

Training Architect
Michael Christian

Linux Academy

1.1 Linux Evolution and Popular Operating Systems

1. **Linux Distributions**

   Distributions:
   A distribution is a collection of components that form a system.

2. **Linux Embedded Systems**

   Embedded Systems:
   A combination of hardware and software for a purpose.

3. **Linux in the Cloud**

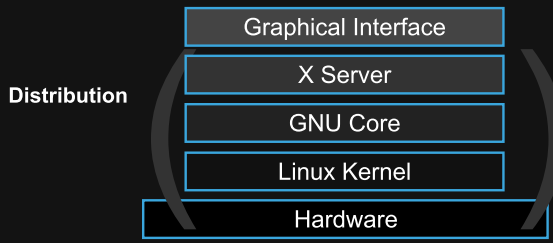
   The Cloud:
   Software and services that run and are available on the internet.

1.1 Linux Evolution and Popular Operating Systems

1

Linux Distributions

Distributions:
A distribution is a collection of components that form a system.



Popular Linux Distributions:

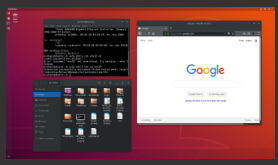
- Red Hat Enterprise Linux
- CentOS
- Ubuntu
- Fedora
- Debian
- openSUSE

1.1 Linux Evolution and Popular Operating Systems

Graphical User Interface:

The Linux GUI is often referred to as a Desktop Environment. Common Linux desktops include Gnome, KDE, Unity, Cinnamon, MATE, and Xfce, though there are many others.

This is the environment that permits you to navigate the operating system using a mouse and folders.



click window to close

Course Navigation

Open Source
Section 1

1.1 Linux Evolution
1.2 Major Applications
1.3 Licensing
1.4 Working in Linux

Finding Your Way
Section 2

Command Line
Section 3

Operating System
Section 4

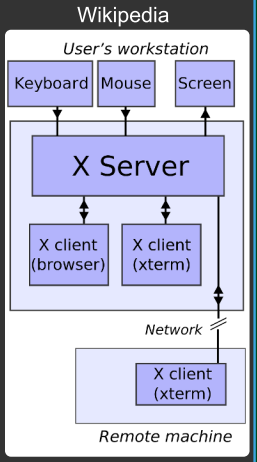
Security
Section 5

1.1 Linux Evolution and Popular Operating Systems

X Server:

The X Server is the display server for the X windows system; a framework for the GUI environment.

X is an architecture agnostic framework for remote graphical user interfaces and input devices. The X server itself was designed to be used over network connections.



click window to close

1.1 Linux Evolution and Popular Operating Systems

GNU Core Utilities:

The GNU Core Utilities are the basic file, shell, and text manipulation utilities of the GNU operating system. These are the foundational **utilities** expected to exist on every operating system.

Examples include: Set and change file permission and ownership, concatenate and write files, exit status, listing directories and files.

click window to close

1.1 Linux Evolution and Popular Operating Systems

Linux Kernel:

The Linux kernel is a free and open-source operating system kernel. The kernel is the framework that connects the application layer to the hardware of a computer.

Within this low-level abstraction process, system calls are made to the kernel. Hardware is incorporated into the file hierarchy through the **/dev** and **/sys** directories, and process information is mapped in **/proc**.

click window to close

1.1 Linux Evolution and Popular Operating Systems

Hardware:

The hardware is the physical device being used as a **compute** resource. This device will include some form of processing capability, as well as short term and long term data storage.

The Linux Distribution is installed as an **operating system** onto the provided hardware.

click window to close

1.1 Linux Evolution and Popular Operating Systems

Red Hat Enterprise Linux:

Red Hat Enterprise Linux (RHEL) is a **commercial** Linux distribution targeted towards the enterprise and commercial market. RHEL uses a paid license model and has support offerings available. This makes it an attractive option for high priority business workloads.

Distro Details

Default Desktop	Gnome
Package Manager	RPM
Init Software	systemd
Release Model	Fixed

clickwindowtoclose

1.1 Linux Evolution and Popular Operating Systems

CentOS Linux:

CentOS is a 100% compatible rebuild of Red Hat Enterprise Linux in compliance with Red Hat's rules regarding redistribution. CentOS is free to use, but does not come with support. This makes CentOS an attractive distribution for users and organizations that need enterprise class stability without the cost of support or certification.

Distro Details

Default Desktop	Gnome
Package Manager	RPM
Init Software	systemd
Release Model	Fixed

click window to close

Course Navigation

Open Source
Section 1

1.1 Linux Evolution
1.2 Major Applications
1.3 Licensing
1.4 Working in Linux

Finding Your Way
Section 2

Command Line
Section 3

Operating System
Section 4

Security
Section 5

1.1 Linux Evolution and Popular Operating Systems

Ubuntu Linux:

Ubuntu is a user-friendly, highly polished desktop distribution with both community and commercial support. Ubuntu's ease of installation and exceptional polish

make it an attractive Linux distribution for beginners and those new to Linux. Ubuntu itself is based on the Debian Linux distribution, but adds extensive retooling and polish.

Distro Details

Default Desktop	Gnome
Package Manager	DEB
Init Software	systemd
Release Model	Rolling

click window to close

1.1 Linux Evolution and Popular Operating Systems

Fedora Linux:

Fedora is developed and maintained by the Fedora Project, and owned by Red Hat. Fedora is a leading edge distribution, often focusing on innovation and integrating the newest technologies, working closely with the upstream communities.

Distro Details

Default Desktop	Gnome
Package Manager	RPM
Init Software	systemd
Release Model	Rolling

click window to close

1.1 Linux Evolution and Popular Operating Systems

Debian Linux:

Debian is a completely free distribution of Linux that is often the foundation for other distributions. It is one of the oldest distributions, dating back to an initial release in 1993. Maintained by its users, Debian supports a large number of hardware architectures, and provides an extensive number (over 40,000) of installation packages.

Distro Details

Default Desktop	Gnome
Package Manager	DEB
Init Software	systemd
Release Model	Rolling

click window to close

1.1 Linux Evolution and Popular Operating Systems

openSUSE Linux:

openSUSE aims to provide a more stable, less experimental system than other Linux distributions. SUSE Linux Enterprise is available as a commercial offering, however openSUSE is community driven and supported. This sponsorship is much like the Red Hat / Fedora relationship.

Distro Details

Default Desktop	KDE
Package Manager	RPM
Init Software	systemd
Release Model	Rolling

click window to close

1.1 Linux Evolution and Popular Operating Systems

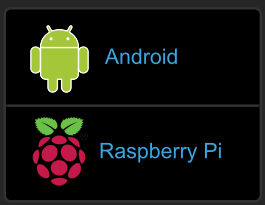
2

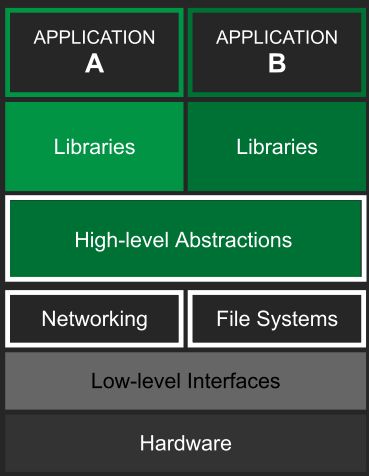
Linux Embedded Systems

Embedded Systems:
A combination of hardware and software for a purpose.

Popular Embedded Linux

Linux is often used in an **embedded** fashion in everything from consumer electronics, industrial automation, networking equipment, and even medical instruments.





1.1 Linux Evolution and Popular Operating Systems

Consumer Electronics:

Operating systems based on the Linux Kernel used in embedded systems for consumer electronics include:

- Set top boxes
- SmartTVs
- In-vehicle entertainment

Many manufacturers, when using system on chips (SoCs), will use their own proprietary version of Linux. The open source kernel forms the foundation of the operating system, however their own **internally-developed drivers** and software control the device.

click window to close

1.1 Linux Evolution and Popular Operating Systems

Network Equipment:

Operating systems based on the Linux Kernel used in embedded systems for networking equipment include:

- Routers
- Switches
- Wireless access points (WAPs)
- Wireless Routers

A standard Linux installation typically has all of the software necessary to be a fully functional router! Netfilter, the framework within the Linux kernel used for firewalls, also provides the hooks and capabilities necessary to track and manage connections across any number of interfaces. Right out of the box it can provide encryption, proxy, and deep packet inspection capabilities.

click window to close

1.1 Linux Evolution and Popular Operating Systems

Android Operating System:

Designed and maintained by Google, Android is a mobile operating system based on a modified version of the Linux kernel. Being primarily designed

for **touchscreen** devices, Android can be found running on smartphones and tablets, as well as watches.

Android heavily leverages Google's proprietary software: **Google Mobile Services**, typically preinstalled with Android. This includes software such as Google **Chrome**, Google **Search**, and **Gmail**.

Android has the **largest** installation base of any operating system.

click window to close

1.1 Linux Evolution and Popular Operating Systems

Raspberry Pi:

The Raspberry Pi is an inexpensive, small sized computer. Often used in the capacity of a **teaching** and **learning** device, the Raspberry Pi is a capable computer that can be leveraged for any number of interesting use cases. Examples range from the controller for an underwater camera drone, to automated plant watering. Additionally, it makes a suitable platform for learning to program in languages like Python.

Raspbian is the Raspberry Pi Foundation's official supported operating system, and can be downloaded and installed for free. Raspbian is based on **Debian**.

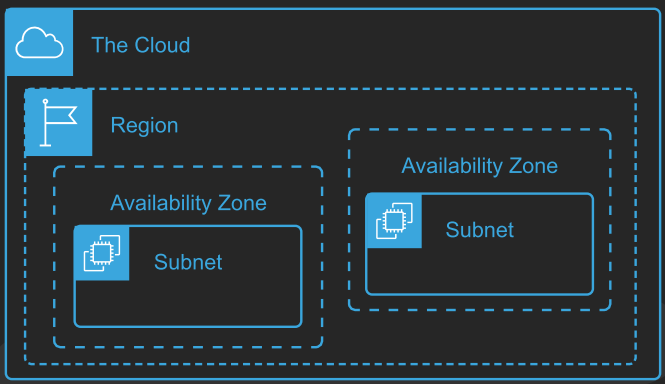
click window to close

1.1 Linux Evolution and Popular Operating Systems

3

Linux in the Cloud

The Cloud:
Software and services that run and are available on the internet.



What is the Cloud?
Put simply, the cloud is a collection of data centers providing compute, application, and storage services over the internet.

1.1 Linux Evolution and Popular Operating Systems

The Cloud:

In this example, we look at the cloud as modeled by **AWS**, or Amazon Web Services. There are many cloud providers, and each may use a different terminology for the layers of infrastructural components that comprise their respective cloud offerings.

This diagram uses basic terms such as **Regions**, **Availability Zones**, and **Subnets**. This is a high level abstraction of the complicated system of systems that enabled the cloud to function as a reliable service.

click window to close

1.1 Linux Evolution and Popular Operating Systems

Region:

As the cloud is hosted in various locations around the world, each of these locations is comprised of a **Region**.

A Region in turn, is a collection of regional data centers. Typically, each region is **independent** of other regions.

Having data stored in regions around the world permits the **shortest** physical path between regional users and the data stored in the cloud, as well as fault tolerance and stability.

click window to close

1.1 Linux Evolution and Popular Operating Systems

Availability Zone:

Housed within Regions are **Availability Zones**. These are the **data centers** that comprise a region. While isolated, Availability Zones are typically connected to each other through low-latency connections.

This provides more a more **local** level of fault tolerance and stability as data may be stored in multiple data center locations within the Region itself.

click window to close

1.1 Linux Evolution and Popular Operating Systems

Subnet:

The subnet is the **local network** instance for the compute resource in the cloud.

There may be multiple computer instances, or **virtual servers**, existing within a subnet, **replicated** into additional Availability Zones in additional Regions. In this manner, the maximum amount of fault tolerance and stability may be achieved.

This is often accompanied by automation mechanisms to create and manage a self-healing infrastructure. **Linux** provides 90% of the public cloud workload.

click window to close

1.2 Major Open-Source Applications

1. **Desktop Applications**
   Desktop: Open-source applications for desktop and office use.
2. **Server Applications**
   Server: Open-source applications that provide client services.
3. **Development Languages**
   Development: Open-source languages for application development.
4. **Package Management Tools**
   Package Management: How installation packages are tracked and managed.

1.2 Major Open-Source Applications

1

Desktop Applications

Desktop:
Open-source applications for desktop and office use.

OpenOffice.org — OpenOffice
LibreOffice — LibreOffice

Office Applications

Web Browser: Firefox

Mail Client: Thunderbird

Image Editor/Drawing: GIMP

1.2 Major Open-Source Applications

OpenOffice:

OpenOffice is developed and maintained by the Apache Software Foundation, and is free to use for any purpose.

OpenOffice is compatible with other major office suites, and contains the following applications:

Writer: Word processor software

Calc: Spreadsheet software

Impress: Presentation software

Draw: Illustration and diagram software

Base: Database software

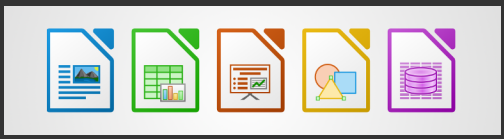
Math: Mathematical equation software

click window to close

1.2 Major Open-Source Applications

LibreOffice:

LibreOffice was developed by developers from the OpenOffice project using the **open source code** from OpenOffice.



Just like OpenOffice, LibreOffice includes applications for word processing (**Writer**), spreadsheets (**Calc**), presentations (**Impress**), vector graphics and flow charts (**Draw**), databases (**Base**), and formula editing (**Math**).

Both OpenOffice and LibreOffice are very similar.

click window to close

Course Navigation

Open Source
Section 1

1.1 Linux Evolution
1.2 Major Applications
1.3 Licensing
1.4 Working in Linux

Finding Your Way
Section 2

Command Line
Section 3

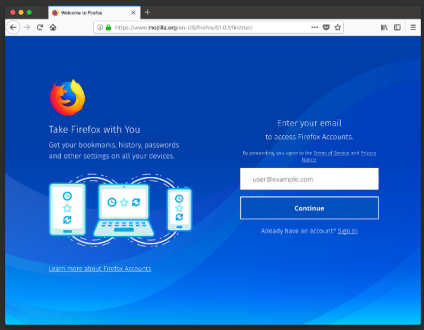
Operating System
Section 4

Security
Section 5

1.2 Major Open-Source Applications

Firefox:

Firefox is an open-source web browser, developed and maintained by the Mozilla Foundation. Created in 2002, Firefox is a popular alternative to proprietary browsers.

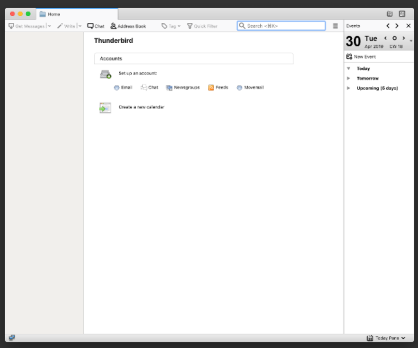


click window to close

1.2 Major Open-Source Applications

Thunderbird:

Mozilla Thunderbird is a free and open-source email and news client, and is capable of functioning as an RSS and chat client. Developed by the Mozilla Foundation, Thunderbird is the default mail client for Ubuntu desktop systems.



click window to close

Course Navigation

Open Source
Section 1

1.1 Linux Evolution
1.2 Major Applications
1.3 Licensing
1.4 Working in Linux

Finding Your Way
Section 2

Command Line
Section 3

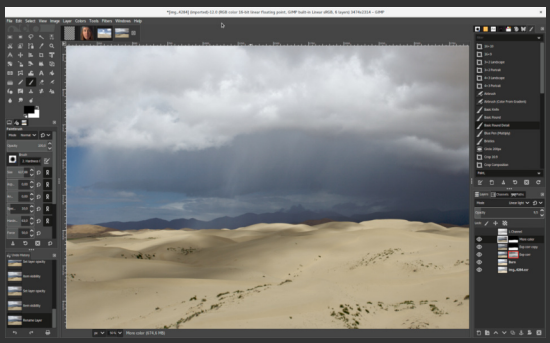
Operating System
Section 4

Security
Section 5

1.2 Major Open-Source Applications

Gimp:

Originally released as the General Image Manipulation Program in 1996, GIMP has evolved into a robust and complete image editing and drawing application.



click window to close

1.2 Major Open-Source Applications

2

Server Applications

Server:
Open-source applications that provide client services.

Web Server Applications:

Apache

NGINX

Database Server Applications:

MySQL

MariaDB

File Sharing Applications:

Samba

NFS

Private Cloud Applications:

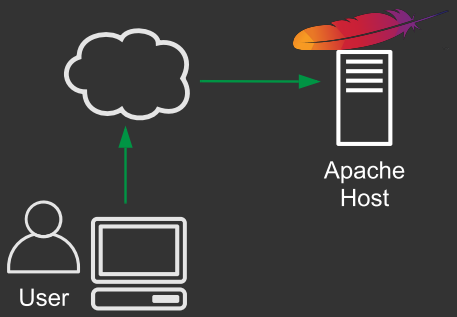
ownCloud

Nextcloud

1.2 Major Open-Source Applications

Apache HTTP Server:

Apache is a highly-popular, free, and open-source web server service application. Apache permits web content to be served by the host, and may use compiled modules to extend the core functionality of the service.

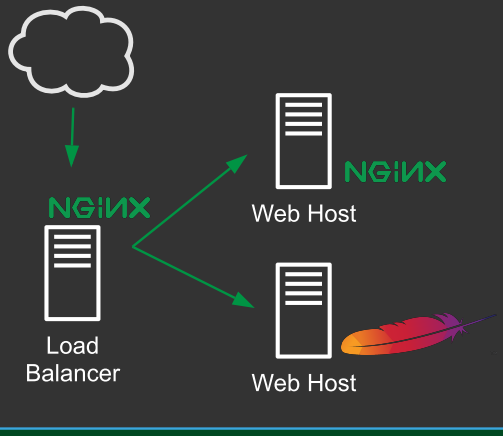


click window to close

1.2 Major Open-Source Applications

NGINX:

NGINX is a web server that can also be used for reverse proxy, load balancing, mail proxy, and HTTP caching.



click window to close

1.2 Major Open-Source Applications

MySQL:

MySQL is an open-source relational database management system. Together with Linux, Apache, and PHP/Perl/Python, it makes up the popular **LAMP** stack used for deploying web sites.

MySQL is used by many database-driven web applications such as WordPress and many websites such as Facebook, Twitter, and YouTube.

click window to close

1.2 Major Open-Source Applications

MariaDB:

MariaDB is a community developed **fork** of MySQL. It is the developers intent for MariaDB to maintain a high level of compatibility between MariaDB and MySQL, while remaining free and open-source software.

MariaDB claims some performance optimizations and speed improvements compared to MySQL, and is a powerful and feature-rich alternative.

Each version of MariaDB is intended to exactly match the corresponding version of MySQL, and compatibility issues are well documented.

click window to close

1.2 Major Open-Source Applications

Samba:

Samba is open-source file sharing software for Linux that permits file sharing with Windows clients through native connectivity using the **Common Internet File System** (CIFS).

Samba performs:

- File and Print Services
- Authentication and Authorization
- Name Resolution
- Service Announcement

The most common use case for Samba is to function as a file server, or as a client to connect to a Windows file server.

click window to close

1.2 Major Open-Source Applications

NFS:

NFS, or Network File System, is a distributed file system protocol, permitting client hosts to access files and directories over the network as local storage.

Use cases for this include shared documents and files, common web application libraries, or even home directories.

click window to close

1.2 Major Open-Source Applications

ownCloud:

ownCloud is client/server suite of applications for creating and using file hosting services. The functionality of ownCloud is similar to Dropbox, however your files are stored on your own connected hardware.

ownCloud supports extensions to permit online document editing, as well as calendar and contact synchronization.

While commercial support is available, the Server Edition of ownCloud is open-source and free.

click window to close

1.2 Major Open-Source Applications

Nextcloud:

Nextcloud

The original developer of ownCloud forked ownCloud to create Nextcloud and continues to actively develop Nextcloud with other members of the original ownCloud team.

While ownCloud core is open-source, the enterprise features aren't. Nextcloud maintains all features as open-source.

In terms of basic functionality, there are few differences between ownCloud and Nextcloud.

click window to close

1.2 Major Open-Source Applications

3

Development Languages

Development:
Open-source languages for application development.

Shell

C

Java

Javascript

Perl

Python

PHP

Course Navigation

Open Source
Section 1

1.1 Linux Evolution
1.2 Major Applications
1.3 Licensing
1.4 Working in Linux

Finding Your Way
Section 2

Command Line
Section 3

Operating System
Section 4

Security
Section 5

1.2 Major Open-Source Applications

Shell:

A shell script is designed to be run by the **command line interpretor** using various scripting languages. Shell scripts are often used to automate strings of commands with some manner of logic applied.

Bash is the most common shell used in Linux; it typically runs as a command processor in a text-based **command line interface**. The ability to read and execute commands from a file is what gives a shell the ability to be used for automated tasks.

click window to close

1.2 Major Open-Source Applications

C:

C is a general-purpose **imperative** programming language designed to be compiled to require minimal runtime support.
C programs can be compiled for a large variety of compute platforms with few changes required to the source code.

C is one of the most **widely used** programming languages with compilers available for the majority of existing computer architectures.

click window to close

Course Navigation

Open Source
Section 1

1.1 Linux Evolution
1.2 Major Applications
1.3 Licensing
1.4 Working in Linux

Finding Your Way
Section 2

Command Line
Section 3

Operating System
Section 4

Security
Section 5

1.2 Major Open-Source Applications

Java:

Java is a class-based, **object oriented** general purpose programming language designed to be compile and run on **any system** supporting the Java runtime via a Java Virtual Machine (JVM). Java is one of the most popular languages in use to date.

The original Java reference compilers, virtual machines, and class libraries were released under a proprietary license by Sun. However, Sun has since released much of Java as free and open-source software (FOSS) under the GNU General Public License (GPL). **OpenJDK** (Open Java Development Kit) is a free and open-source implementation of the Java programming language.

click window to close

Course Navigation

Open Source
Section 1

1.1 Linux Evolution
1.2 Major Applications
1.3 Licensing
1.4 Working in Linux

Finding Your Way
Section 2

Command Line
Section 3

Operating System
Section 4

Security
Section 5

1.2 Major Open-Source Applications

Javascript:

HTML, CSS, and Javascript make up the core technologies of the World Wide Web. Originally implemented as a client side technology, JS engines are now available in many types of host software.

Most major web browsers have a dedicated JavaScript engine for JS execution, as it is a technology that enables **interactive web pages**.

Java and JavaScript are two different and distinct languages.

click window to close

1.2 Major Open-Source Applications

Perl:

Originally developed in 1987 as a general purpose Unix scripting language, Perl has become a high-level, general-purpose programming language. Perl usually refers to Perl 5, but it may also be used to refer to Perl 6.

Perl supports both **procedural** and **object-oriented** programming, and is highly extensible with over 20,000 third-party modules available. Due to its significant flexibility, Perl is sometimes known as the "Duct Tape of the Internet".

click window to close

1.2 Major Open-Source Applications

Python:

Python is an interpreted, general-purpose programming language similar to Perl but with different principles in terms of flexibility. Perl has many ways to perform a single task, where as Python focuses on a single method.

In terms of comparisons, Python source code is generally regarded as easier to read due to structuring requirements.

Python has extensive object-oriented programming support with clean and consistent syntax.

click window to close

Course Navigation

Open Source
Section 1

1.1 Linux Evolution
1.2 Major Applications
1.3 Licensing
1.4 Working in Linux

Finding Your Way
Section 2

Command Line
Section 3

Operating System
Section 4

Security
Section 5

1.2 Major Open-Source Applications

PHP:

PHP: Hypertext Processor or PHP, is a programming language originally designed for **web development**.

PHP code may be embedded into HTML, used in combination with various web frameworks and template systems, or executed on the command line.

The standard PHP interpreter is free and released under the PHP license, a license designed to encourage widespread adoption of the source code.

click window to close

1.2 Major Open-Source Applications

4

Package Management Tools

Package Management: How installation packages are tracked and managed.

dpkg: Debian Package

apt-get: Advanced Package Tool

rpm: Red Hat Package Manager

yum: Yellowdog Updater, Modified

1.2 Major Open-Source Applications

dpkg:

dpkg, short for Debian Package, is a package archive format, containing some package metadata. DEB files are package installation files for Debian based Linux distributions.

Package files are designed to to make the installation of programs easier by containing the files necessary for the installation, as well as some metadata around **dependencies**.

Debian based systems that rely on dpkg files for package installation can take advantage of APT (Advanced Package Tool) for managing packages and their dependencies.

click window to close

1.2 Major Open-Source Applications

apt-get:

Advanced Package Tool (APT) is a free and open source interface that works with core libraries to handle the installation and removal of software packages on Debian-based system. APT simplifies installation by automating the retrieval and validation of software packages as well as their dependencies.

APT permits the user to easily search for and install packages from **repositories**, directories containing software packages and an index. These repositories can be local, or accessed over a network.

Various GUI front ends are available for working with APT, such as Synaptic, or the Ubuntu Software Center.

click window to close

1.2 Major Open-Source Applications

rpm:

While originally used by Red Hat systems, RPM is a package management system used by a wide variety of Linux distributions.

While dpkg uses .deb files, RPM uses .rpm files. And just like DEB, an RPM file is an **archive** of package installation files as well as metadata containing dependencies and installation paths.

Most RPM files are binary RPMs containing the compiled version of the software to install. However, SRPMs (**Source RPMs**) are available, containing the source code for local compilation.

And just as dpkg systems may use APT for package management, RPM-based systems may use **YUM**.

click window to close

1.2 Major Open-Source Applications

yum:

Yum, or the Yellowdog Updater, Modified, is a free and open-source command line package management utility for **RPM-based** Linux systems. Just as APT for Debian -based systems, YUM allows for automated updates and package dependency management.

Also like APT, YUM makes use of software **repositories** that may be locally accessed or available on a network.

A rewrite of YUM named **DNF** has replaced YUM as the default package manager for Fedora systems. DNF was created to improve upon YUM in several ways: better performance, better dependency resolution, and easier integration with other applications.

click window to close

1.3 Open-Source Software and Licensing

1. **Philosophy**
   Open-Source Philosophy: Open-source philosophy and design
2. **Licensing**
   Open-Source Licensing: Open-source licensing models and types
3. **FSF and OSI**
   FSF and OSI: Free Software Foundation and Open Source Initiative

1.3 Open-Source Software and Licensing

1

Philosophy

Open-Source Philosophy: Open-source philosophy and design

The basic premise of open-source software is that the **source code** is available for anyone to freely view, use, modify, or redistribute.

Forking

Forking occurs when software developers use the source code from one project, to start an entirely **new** parallel project.

Popular Examples

OpenOffice.org LibreOffice

ownCloud Nextcloud MySQL MariaDB

A Philosophical Crossroads

Allow users to freely copy, distribute, and change the software

Permissive

No restrictions on licensing derivative work

Copyleft

Derivative work **must** use the **same license** as the original software

1.3 Open-Source Software and Licensing

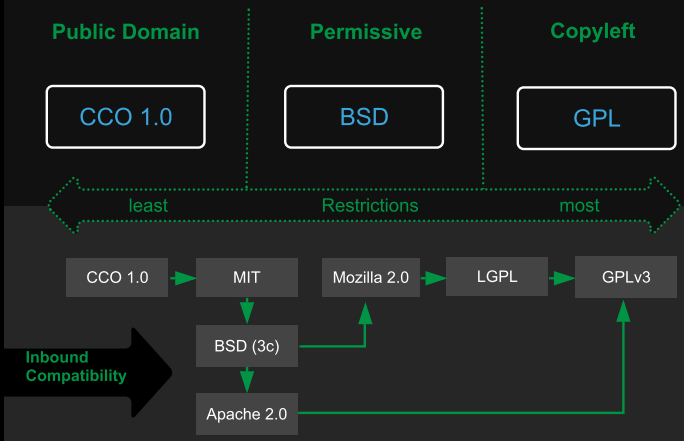
2

Licensing

Open-Source Licensing:
Open-source licensing models and types

Open-source licensing provides **rules** and **guidelines** for how the work may be used, permitting others to contribute without first seeking permission.

License Examples



1.3 Open-Source Software and Licensing

CCO 1.0 Universal:

The Creative Commons 1.0 Universal Public Domain Dedication is a declaration that all works are in the public domain.

Excerpt:

*The person who associated a work with this deed has **dedicated** the work to the public domain by waiving all of his or her rights to the work worldwide under copyright law, including all related and neighboring rights, to the extent allowed by law.*

You can copy, modify, distribute and perform the work, even for commercial purposes, all without asking permission.

Creative Commons — CC0 1.0 Universal

click window to close

Course Navigation

Open Source
Section 1

1.1 Linux Evolution
1.2 Major Applications
1.3 Licensing
1.4 Working in Linux

Finding Your Way
Section 2

Command Line
Section 3

Operating System
Section 4

Security
Section 5

1.3 Open-Source Software and Licensing

BSD:

BSD licenses are a segment of permissive software licenses, with minimal restrictions imposed on usage and distribution.

BSD licenses typically contain 4 or fewer of the following **clauses**:

1. Redistributions of **source code** must retain the above copyright notice, this list of conditions, and the following disclaimer.
2. Redistributions in **binary** form must reproduce the above copyright notice, this list of conditions, and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All **advertising** materials mentioning features or use of this software must display the following **acknowledgement**:
   This product includes software developed by the <organization>.
4. Neither the name of the <organization> nor the names of its contributors **may be used to endorse** or **promote** products derived from this software without specific prior written permission.

click window to close

Course Navigation

Open Source
Section 1

1.1 Linux Evolution
1.2 Major Applications
1.3 Licensing
1.4 Working in Linux

Finding Your Way
Section 2

Command Line
Section 3

Operating System
Section 4

Security
Section 5

1.3 Open-Source Software and Licensing

GPL:

The GPL, or GNU General Public License, is a widely-used copyleft software license.

Being a **copyleft** license, derivative work may only be distributed under the **same license terms**.

This license permits:

- commercial use
- modification
- distribution

restricts:

- sublicensing

and must:

- include the original work
- stated changes

click window to close

1.3 Open-Source Software and Licensing

3

FSF and OSI

FSF and OSI:
Free Software Foundation and Open Source Initiative

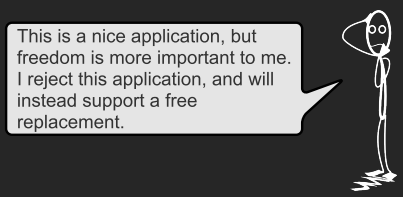
The Open Source Initiative and the Free Software Foundation are two movements that place approvals on open-source software licenses.

Open Source Initiative

Individuals from both sides often work with each other on free, open-source projects. Both support free, modifiable software and openly-available source code.

FOSS vs FLOSS

Free Software Foundation



The primary difference in philosophies stems from a lack of focus on the practical benefits of the software, and more on the ethics of rights restrictions regarding proprietary software.

Course Navigation

Open Source
Section 1

1.1 Linux Evolution
1.2 Major Applications
1.3 Licensing
1.4 Working in Linux

Finding Your Way
Section 2

Command Line
Section 3

Operating System
Section 4

Security
Section 5

1.3 Open-Source Software and Licensing

FOSS versus FLOSS:

The difference between FOSS and FLOSS is largely philosophical; the practical difference is that FLOSS implies slightly more restrictions in terms of monetization. In terms of semantical differences:

FOSS: Free and Open Source Software
The word **free** here is generally regarded in the context of **price**.

FLOSS: Free/Libre/Open Source Software
The word **free** here is a reference to **freedom**, and this designation attempts to the make the meaning more clear.

Thus, if you want to be neutral between free software and open source, and clear about them, the way to achieve that is to say "FLOSS," not "FOSS." -- *Richard Stallman*

click window to close

1.4 ICT Skills and Working in Linux

1. **Desktop Skills**
   Desktop Skills: Using the Linux Desktop, config options, web usage, and privacy.
2. **Getting to the CLI**
   Getting to the Command Line: How to access the Linux command line interface.
3. **Industry Uses**
   Industry Uses of Linux: How Linux is used in virtualization and cloud computing.

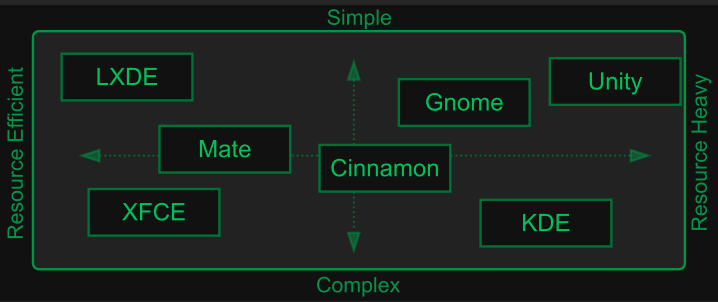
1.4 ICT Skills and Working in Linux

1

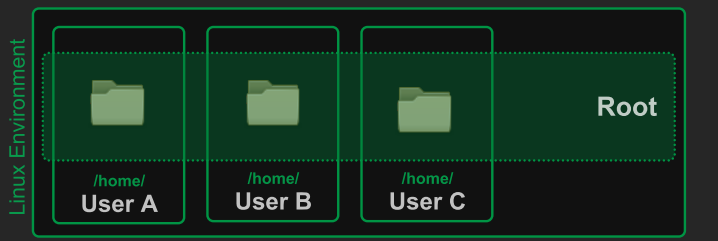
Desktop Skills

Desktop Skills:
Using the Linux Desktop, config options, web usage, and privacy.

Popular Linux Desktop Environments



Userspace and Privacy



1.4 ICT Skills and Working in Linux

2

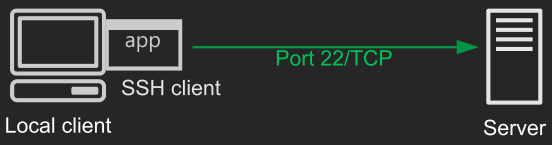
Getting to the CLI

Getting to the Command Line: How to access the Linux command line interface.

When using Linux as a server environment, often the only interface to the system is the **Command Line**.

```
[cloud_user@ip-10-0-1-10 ~]$  ls
file1 file2 file3 file4
[cloud_user@ip-10-0-1-10 ~]$
```

Remote Connections



1.4 ICT Skills and Working in Linux

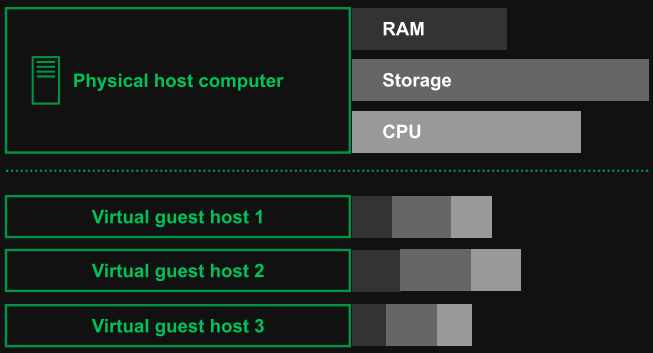
3

Industry Uses

Industry Uses of Linux:
How Linux is used in virtualization and cloud computing.

Virtualization and **cloud computing** have changed the shape of modern compute workloads, and **Linux** has been at the **forefront** of this digital transformation.

Virtualization



Linux in the Cloud (1.1 Review)

2.1 Command Line Basics

1. **Basic Shell**
   Basic Shell: How the shell interprets commands for translation and execution.
2. **Command Line Syntax**
   Command Line Syntax: Basic command line syntax and notation.
3. **Variables**
   Variables: Using variables to store values for easy reference.
4. **Quoting**
   Quoting: Preserving input that contains special characters or spaces.

2.1 Command Line Basics

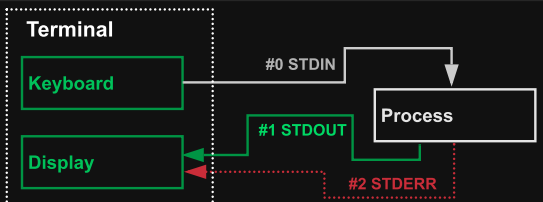
1

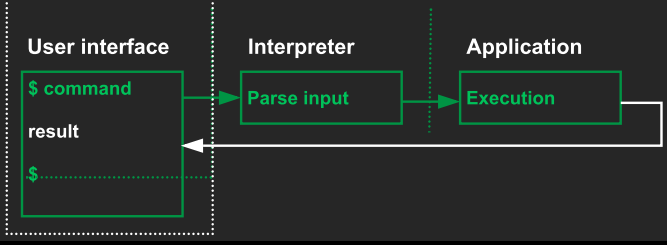
Basic Shell

Basic Shell:
How the shell interprets commands for translation and execution.

The standard Linux shell (BASH) is both a command line interpreter and a programming language.

```
[cloud_user@ip-10-0-1-10 ~]$ ls
file1 file2 file3 file4
[cloud_user@ip-10-0-1-10 ~]$
```





2.1 Command Line Basics

Command Prompt:

```
[cloud_user@ip-10-0-1-10 ~]$
```

The command prompt, is a short text string at the beginning of the command line. Linux command prompts typically show the current **user**, the current **host**, and the correct working **directory**.

The command prompt is easily modified to show more or less information as desired. The **dollar sign** (**$**) at the end of the prompt string denotes that the current user user is unprivileged, and that the interface is ready to receive input.

This input is sent to the interpretor to be parsed and executed. **BASH**, or the Bourne Again Shell, is the most common interpretor, however, there are others available, some of which do not use the dollar sign.

click window to close

2.1 Command Line Basics

Commands:

```
$ ls
```

Commands are entered as text after the command line prompt, are parsed by the interpreter, and executed by the shell as a **process**.

Commands are entered as text into the input stream known as Standard In (**STDIN**). These commands adhere to a **syntax** as defined by the interpreter or current shell. The commands entered may be as simple as invoking a single application or utility; or as complex as **redirecting** the **output** from one utility to another, while iterating through a logical loop.

The ability to redirect output, as well as apply logic, to a large number of useful GNU utilities and applications makes the command line a powerful tool for system administration and **automation**.

click window to close

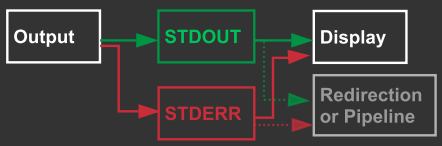
2.1 Command Line Basics

Command Output:

```
file1   file2   file3   file4
```

The output of the entered command is displayed on the line following the command itself. Commands are sent for execution by using [ENTER].

This output will be sent to one of two streams: Standard Out (**STDOUT**), or Standard Error (**STDERR**). Both streams will print to the screen by default, however, the output may be changed with **redirection** or a **pipeline**.



click window to close

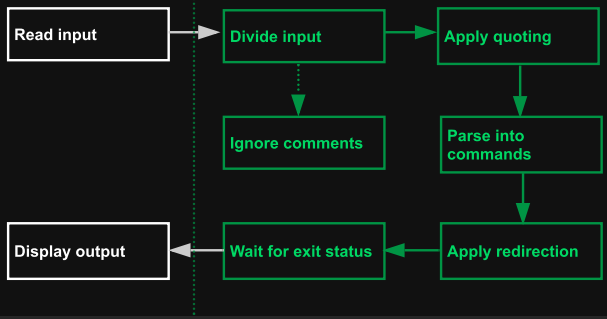
2.1 Command Line Basics

2

Command Line Syntax

Command Line Syntax:
Basic command line syntax and notation.

As the shell input is read, it follows a sequence of operations, ignoring comments (#) and divvying the input into words and operators.



Example Command Syntax

```
ls  [ OPTION ]...  [ FILE ]...
     -a, --all
          do not ignore entries starting with .
     -h, --human-readable
          with -l and -s, print sizes like 1K 234M 2G etc.
```

2.1 Command Line Basics

3

Variables

Variables:
Using variables to store values for easy reference.

A bash variable may contain a **number**, a **character**, or a **string** of characters. These variables do not need to be declared, nor are there data types. They are simply assigned.

Assigning a Variable

```
[cloud_user@ip-10-0-1-10 ~]$ test_variable="This is a variable"
[cloud_user@ip-10-0-1-10 ~]$ echo $test_variable
This is a variable
[cloud_user@ip-10-0-1-10 ~]$
```

Bourne Shell Variables

$HOME the current user's home directory
$PS1 the primary prompt string
$PATH a colon-separated list of directories where the shell looks for commands

2.1 Command Line Basics

4

Quoting

Quoting:
Preserving input that contains special characters or spaces.

Quoting is used to **disable** special treatment of certain characters and words, as well as to prevent parameter expansion and **preserve** what is quoted.

Methods

Escape Character
A non-quoted backslash **** is the bash escape character and preserves the literal value of the **next following character**, with the single exception of *newline*.

Single Quotes
Single quotes **'** preserve the literal value of **every character** contained within the quotes, including the escape character.

Double Quotes
Double quotes **"** preserve the literal value of **most characters** contained within the quotes, exceptions include **$** (for variables), **'**(for single quoting), ****(for escaping a character).

Examples

2.1 Command Line Basics

Shell Quoting Examples:

Escape Character

```
[cloud_user@ip-10-0-1-10 ~]$ var="These are quotes(\")"
[cloud_user@ip-10-0-1-10 ~]$ echo $var
These are quotes(")
```

Single Quotes

```
[cloud_user@ip-10-0-1-10 ~]$ var='These are quotes(")'
[cloud_user@ip-10-0-1-10 ~]$ echo $var
These are quotes(")
[cloud_user@ip-10-0-1-10 ~]$ var='The PS1 variable is $PS1'
[cloud_user@ip-10-0-1-10 ~]$ echo $var
The PS1 variable is $PS1
```

Double Quotes

```
[cloud_user@ip-10-0-1-10 ~]$ var="These are quotes(")"
bash: syntax error near unexpected token `)'
[cloud_user@ip-10-0-1-10 ~]$ var="The PS1 variable is $PS1"
[cloud_user@ip-10-0-1-10 ~]$ echo $var
The PS1 variable is [\u@\h \W]\$
```

click window to close

2.2 Using the Command Line to Get Help

1

Man Pages

Man Pages:
Traditional package documentation for application usage.

2

Info Pages

Info Pages:
Additional documentation with more robust capability and detail.

2.2 Using the Command Line to Get Help

1

Man Pages

Man Pages:
Traditional package documentation for application usage.

Manual pages are the traditional form of software documentation found on UNIX systems. They are included with the software they document.

Usage

The man page for a particular command is invoked by preceding the command with **man**.

```
man <command>
```

Sections

NAME: (Required) Program or function name(s) followed by a terse descriptions of functionality.

SYNOPSIS: A short overview of available options

DESCRIPTION: Detailed information of arguments and options.

EXAMPLES: Common usage examples.

2.2 Using the Command Line to Get Help

2

Info Pages

Info Pages:
Additional documentation with more robust capability and detail.

Info pages normally provide more **detailed** information about a command than its respective man page. Additionally, info uses a a structure for **linking** pages together, and may be assembled into a larger collection.

Usage

The info page for a particular command is invoked by preceding the command with **info**.

```
info <command>
```

If no info page exists, info can pull documentation from the **man page**.

2.3 Using Directories and Listing Files

1. **Files and Directories**
   Files and Directories: The hierarchy of storage on the Linux operating system.
2. **Hidden Files and Directories**
   Hidden Files and Directories: Files and directories that are hidden from basic listing.
3. **Home Directories**
   Home Directories: User home directories containing the user's files and directories.
4. **Absolute and Relative Paths**
   Absolute and Relative Paths: The path to the unique location of a file or directory.

2.3 Using Directories and Listing Files

1

Files and Directories

Files and Directories:
The hierarchy of storage on the Linux operating system.

The Filesystem Hierarchy Standard (FHS) defines the structure of the file systems on Linux.

The Root Filesystem | Moving Around

/	the root directory
/bin	user binaries
/boot	static boot files
/dev	device files
/etc	configuration files
/home	home directories
/lib	shared libraries
/mnt	temporary mount points
/opt	optional packages
/proc	kernel and process files
/root	root user home directory
/run	application state files
/sbin	system administration binaries
/srv	service data
/tmp	temporary files
/usr	user binaries
/var	variable data files

2.3 Using Directories and Listing Files

Moving around:

Common commands for moving around a Linux file system:

Get current location:
`pwd`

Change directory:
`cd <NEW LOCATION>`

Go back to last location:
`cd -`

Go up one level:
`cd ..`

Go up two levels:
`cd ../..`

click window to close

2.3 Using Directories and Listing Files

1

Hidden Files and Directories

Hidden Files and Directories: Files and directories that are hidden from basic listing.

Files and directories in Linux may be hidden by preceding their name with a period.

Hidden Files

```
[cloud_user@ip-10-0-1-10 ~]$ ls
file1 file2 file3 file4
[cloud_user@ip-10-0-1-10 ~]$  touch .file5
[cloud_user@ip-10-0-1-10 ~]$ ls
file1  file2  file3  file4
[cloud_user@ip-10-0-1-10 ~]$ ls -a
.   ..  .file5 file1  file2  file3  file4
[cloud_user@ip-10-0-1-10 ~]$
```

Hidden files may be listed via **ls** using the **-a** option, to show **all** files.

2.3 Using Directories and Listing Files

1

Home Directories

Home Directories:
User home directories containing the user's files and directories.

A home directory is typically created for every **ordinary** user on a Linux system under **/home**.

Home Directory

```
[cloud_user@ip-10-0-1-10 ~]$ pwd
/home/cloud_user
[cloud_user@ip-10-0-1-10 ~]$  cd
[cloud_user@ip-10-0-1-10 ~]$ pwd
/home/cloud_user
[cloud_user@ip-10-0-1-10 ~]$ cd ~
[cloud_user@ip-10-0-1-10 ~]$ pwd
/home/cloud_user
[cloud_user@ip-10-0-1-10 ~]$ cd $HOME
[cloud_user@ip-10-0-1-10 ~]$ pwd
/home/cloud_user
```

cd (change directory) without a specified path

~ used to represent user's home directory

$HOME environment variable used to store home directory path

2.3 Using Directories and Listing Files

1

Absolute and Relative Paths

Absolute and Relative Paths: The path to the unique location of a file or directory.

The path to a file or directory location may be specified as an absolute path, starting with a */*, or a relative path.

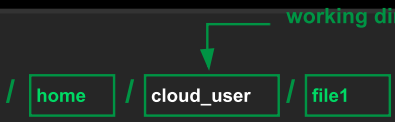
```
[cloud_user@ip-10-0-1-10 ~]$ pwd
/home/cloud_user
```

Relative Path

```
[cloud_user@ip-10-0-1-10 ~]$  cat file1
Hello world.
```

Absolute Path

```
[cloud_user@ip-10-0-1-10 ~]$  cat
/home/cloud_user/file1
Hello world.
```



Relative path: file1

Absolute path: /home/cloud_user/file1

2.4 Creating, Moving, and Deleting Files

1. **Creating, Moving, Deleting**
   Creating, Moving, Deleting: Creating, moving, and deleting files and directories.

2. **Case Sensitivity**
   Case Sensitivity : Working with a case-sensitive file system.

3. **Simple Globbing**
   Simple Globbing: Using partial matching to work with groups of files and directories.

2.4 Creating, Moving, and Deleting Files

1

Creating, Moving, Deleting

Creating, Moving, Deleting:
Creating, moving, and deleting files and directories.

Working with **directories**:

Create a new directory
mkdir <NAME>

Copy a directory
cp -r <SOURCE> <DESTINATION>

Move a directory
mv <SOURCE> <DESTINATION>

Delete a directory
rm -r <DIRECTORY>

Working with **files**:

Create a new file
touch <NAME>

Copy a file
cp <SOURCE> <DESTINATION>

Move a file
mv <FILE> <NEW LOCATION>

Delete a file
rm <FILE>

2.4 Creating, Moving, and Deleting Files

2

Case Sensitivity

Case Sensitivity :
Working with a case-sensitive file system.

Most of the common Linux file systems are case sensitive; this is something to keep in mind when creating and moving through directories and files.

Using Case Sensitivity

Lower-case and upper-case letters have different ASCII representation, thus:

touch newfile

and

touch Newfile

Will create **two different files**.

2.4 Creating, Moving, and Deleting Files

3

Simple Globbing

Simple Globbing:
Using partial matching to work with groups of files and directories.

Globbing is primarily used to match patterns in filenames or text by using a **wildcard character** to create the pattern.

? Question mark - match any single character

* Asterisk - match any number of character(s)

[] Brackets - match character from a range

^ Caret - used to match starting character

$ Dollar sign - used to match ending character

{} Curly brace - used to match more than one pattern

| Pipe - used for applying more than one condition

2.4 Creating, Moving, and Deleting Files

? - Question Mark:

Used to match any single character:

```
[cloud_user@ip-10-0-1-10 ~]$ ls
file1 file2 file3 file4
[cloud_user@ip-10-0-1-10 ~]$ ls file?
file1 file2 file3 file4
[cloud_user@ip-10-0-1-10 ~]$ ls ?????
file1 file2 file3 file4
[cloud_user@ip-10-0-1-10 ~]$ ls ????1
file1
[cloud_user@ip-10-0-1-10 ~]$
```

click window to close

2.4 Creating, Moving, and Deleting Files

* - Asterisk

An asterisk is used to match any number of characters:

```
[cloud_user@ip-10-0-1-10 ~]$ ls
file1 file2 file3 file4
[cloud_user@ip-10-0-1-10 ~]$ ls file*
file1 file2 file3 file4
[cloud_user@ip-10-0-1-10 ~]$ ls *
file1 file2 file3 file4
[cloud_user@ip-10-0-1-10 ~]$ ls *1
file1
[cloud_user@ip-10-0-1-10 ~]$
```

click window to close

2.4 Creating, Moving, and Deleting Files

[] - Brackets:

Brackets are used to match a character from a range.

```
[cloud_user@ip-10-0-1-10 ~]$ ls
file1 file2 file3 file4
[cloud_user@ip-10-0-1-10 ~]$ ls *[1-4]
file1 file2 file3 file4
[cloud_user@ip-10-0-1-10 ~]$ ls *[2-3]
file2 file3
[cloud_user@ip-10-0-1-10 ~]$ ls *[[:digit:]]
file1 file2 file3 file4
[cloud_user@ip-10-0-1-10 ~]$
```

[[:upper:]] or **[A-Z]** - match upper-case character
[[:lower:]] or **[a-z]** - match lower-case character
[[:digit:]] or **[0-9]** - match digits
[[:alpha:]] or **[a-zA-Z]** - match either case character
[[:alphanum:]] or **[a-zA-Z0-9]** - match alphanumeric

click window to close

3.1 Archiving Files on the Command Line

1

Archiving Files and Directories

Archiving Files and Directories: Creating archives of files and directories.

2

Archives and Compression

Archives and Compression: Using compression to reduce storage consumption.

3.1 Archiving Files on the Command Line

1

Archiving Files and Directories

Archiving Files and Directories: Creating archives of files and directories.

Archiving is the process of combining multiple files and/or directories into a single file. This is generally done as part of a backup process. The most common utility for creating and working with archives in Linux is:

tar - Tape Archive

Options
-c create archive
-x extract archive
-r append to an archive
-t list the contents of an archive
-f read from or write to a file

Create an archive:
```
[cloud_user@... ~]$ ls
file1 file2 file3 file4
[cloud_user@... ~]$ tar cf archive.tar *
```
Extract an archive:
```
[cloud_user@... ~]$ tar xf archive.tar
```

3.1 Archiving Files on the Command Line

2

Archives and Compression

Archives and Compression: Using compression to reduce storage consumption.

Compression is the process of reducing the amount of storage that files or archives consume. Compression is typically used during archiving to reduce the storage space needed for archives. The three most common compression algorithms used by Linux are:

gzip - the default compression used by tar via (-z), a good balance of speed and size reduction.

bzip2 - an alternative compression algorithm, typically slower than gzip due to higher compression.

zip - the algorithm used by the **zip** command, and all-in-one compression and archiving utility popular with other operating systems.

3.2 Searching and Extracting Data from Files

1. **Command Line Pipes**
   Command Line Pipes: Piping output from one command into another.

2. **I/O Redirection**
   I/O Redirection: Redirecting output to or from a file.

3. **Basic Regular Expressions**
   Basic Regular Expressions: Using regular expressions for pattern matching.

3.2 Searching and Extracting Data from Files

1

Command Line Pipes

Command Line Pipes:
Piping output from one command into another.

Piping is the process of using the output of one command as the input for another. This is one of the most common and powerful tools in the command line.

| — The pipe symbol
grep — Utility that searches any given input files, selecting lines that match one or more patterns

Example:

```
[cloud_user@... ~]$ ls
file1 file2 file3 file4
[cloud_user@... ~]$ ls | grep 1
file1
[cloud_user@... ~]$ pwd
/home/cloud_user
[cloud_user@... ~]$ pwd | grep cloud
/home/cloud_user
```

3.2 Searching and Extracting Data from Files

2

I/O Redirection

I/O Redirection:
Redirecting output to or from a file.

I/O redirection may be used to feed input to a command from a file or to send the output of a command to a file.

[COMMAND] < [FILE] — Read input from a file

```
Example:
[cloud_user@... ~]$ cat file1
This
is a test
[cloud_user@... ~]$ grep test < file1
is a test
```

[COMMAND] > [FILE] — Send output to a file

```
Example:
[cloud_user@... ~]$ ls > file5
[cloud_user@... ~]$ cat file5
archive.tar file1 file2 file3 file4 file6
```

[COMMAND] >> [FILE] — Append output to a file

3.2 Searching and Extracting Data from Files

3

Basic Regular Expressions

Basic Regular Expressions:
Using regular expressions for pattern matching.

Regular expressions (or *regex*) are used to match patterns in text, similar to **globbing**.

Match start of line:
"^Apple"

Match end of line:
"Apple$"

Match start and end of line:
"^Apple$"

Match either string or character:
"Apple|Ball"

Match A, followed by zero or more p's, followed by le:
"Ap*le"

Match A, followed by one or more p's, followed by le:
"Ap+le"

Match A, followed by maybe a p, followed by le:
"Ap?le"

Match Ap, followed by a letter between p and z, followed by le:
"Ap[p-z]le"

3.3 Turning Commands into a Script

1

Basic Shell Scripting

Basic Shell Scripting:
The essentials of creating and running a shell script.

2

Common Text Editors

Common Text Editors:
CLI applications for working with text files.

3.3 Turning Commands into a Script

1

Basic Shell Scripting

Basic Shell Scripting:
The essentials of creating and running a shell script.

Anatomy of a Shell Script

```
#!/bin/bash

# These are comments

echo "Please enter your name: "
read name
echo -e "Hello $name!\n"
```

!#/bin/bash
This top line defines the interpretor to be used by the shell script.

 # Comments
Comments are not executed as part of the shell script and are used for documentation within the script itself.

If Statements | For Loops

3.3 Turning Commands into a Script

If Statements:

```
if [ <some test> ]
then
 <commands>
fi
```

! [expression]	The expression is false.
-n [string]	The length of the string is greater than zero.
-z [string]	The length of the string is zero (empty).
string1 = string2	string1 is equal to string2.
string1 != string2	string1 is not equal to string2.
integer1 -eq integer2	integer1 is equal to integer2.
integer1 -gt integer2	integer1 is greater than integer2.
integer1 -lt integer2	integer1 is less than integer2.
-d [file]	The file exists and is a directory.
-e [file]	The file exists.
-r [file]	The file exists, and the read permission is granted.
-s [file]	The file exists, and its size is greater than zero.
-w [file]	The file exists, and the write permission is granted.
-x [file]	The file exists, and the execute permission is granted.

click window to close

3.3 Turning Commands into a Script

For Loops:

```
for <variable> in <list>;
   do <command(s)>;
done
```

Examples:

```
for i in 1 2 3;  do echo $i; done
1
2
3

for i in $(ls); do echo $i; done
file1
file2
file3
file5
file5
```

click window to close

3.3 Turning Commands into a Script

2

Common Text Editors

Common Text Editors:
CLI applications for working with text files.

vim

Moving Around:
gg - Go to the first line of the file
G - Go to the last line of the file
10G - Go to the 10th line of the file

Inserting Text:
i - Insert at cursor
I - Insert at start of line
o - Append line under cursor

Exiting:
:w - Write file
:q - Quit vim
:wq - Write and quit

nano

Exiting:
CTRL + X - Exit, will prompt to save

Saving:
CTRL + o - Exit, will prompt to save

4.1 Choosing an OS

1

OS Differences

OS Differences:
Differences between Microsoft Windows, macOS, and Linux.

2

Distribution Lifecycle

Distribution Lifecycle:
Distribution lifecycle management and release models.

4.1 Choosing an OS

1

OS Differences

OS Differences:
Differences between Microsoft Windows, macOS, and Linux.

Comparison

	Linux	macOS	Windows
Open Source	Yes	No	No
Purchase Price	Free*	Free*	Varies
Supported Hardware	Excellent	Proprietary	Good
Shell Interpreter	Bash	Bash	PowerShell
GUI	Multiple	macOS	Windows

* Enterprise licensing and support available
* Only available on proprietary hardware

4.1 Choosing an OS

2

Distribution Lifecycle

Distribution Lifecycle:
Distribution lifecycle management and release models.

Release Models

Stability

Cutting Edge

Standard Distribution Release

Also known as a point release (or *stable release*), a standard distribution release is made available after a development period. During this development period, all software is **updated to a version** and then **frozen** and tested to verify that all software versions work well together. Updates are released to address important bugs and security concerns.

Rolling Release

The distribution is always kept up to date using **small** and **frequent updates** to the core of the OS. New versions of software are added as they arrive.

4.2 Computer Hardware

1

Hardware

Hardware:
The physical components used for computing.

4.2 Computer Hardware

1

Hardware

Hardware:
The physical components used for computing.

Computing Hardware

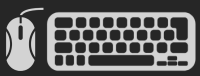
Drivers

Central Processing Unit (**CPU**)
Processes computer functions and performs calculations

Random Access Memory (**RAM**)
High-performance, **volatile** storage

Secondary Storage (**HDD/SSD/DVD**)
Persistent storage for data not currently in use

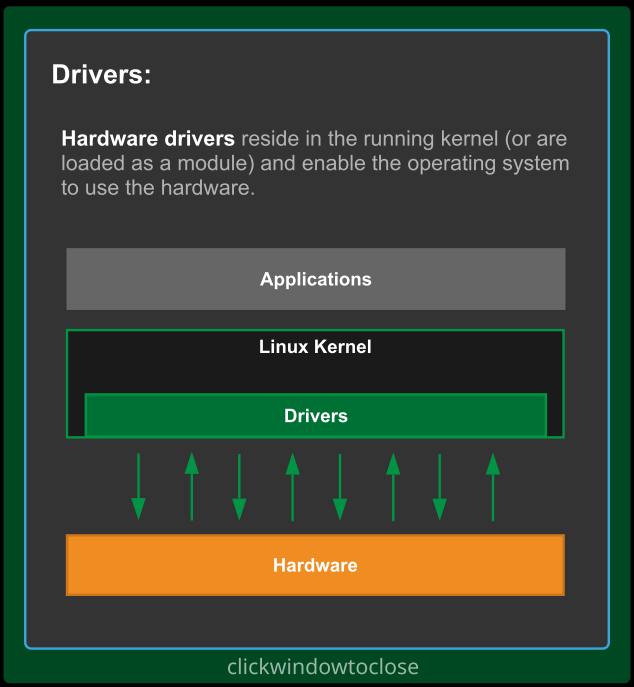
Network Interface Card (**NIC**)
Permits **connections** to a network



Input Devices (**Mouse/Keyboard**)
Send data **into** the computer via human interaction

Output Devices (**Monitor**)
Send information **from** the computer to the user

4.2 Computer Hardware



4.3 Where Data Is Stored

1. **Programs and Configuration**
   Programs and Configuration: Common locations for system and configuration data.
2. **Processes**
   Processes: Data used by running processes.
3. **System Messaging**
   System Messaging: Viewing and accessing system messaging.
4. **Logging**
   Logging: Common locations for system and application log data.

4.3 Where Data Is Stored

1

Programs and Configuration

Programs and Configuration: Common locations for system and configuration data.

System Configuration

System Boot Configuration: /boot
Contains boot loader configuration files and parameters, Linux kernel, and initial RAM disk.

Partition Mount Points: /etc/fstab
Contains a list of partitions to mount automatically and where they mount in the filesystem.

User Attributes: /etc/passwd
Contains a list of local users and their attributes.

Groups: /etc/group
Contains a list of local users and attributes.

Hosts File: /etc/hosts
Contains a list of IP addresses and the hostname we want the system to associate with them.

Application Configuration: /etc/<APPLICATION>
Applications place their respective configuration files in the /etc directory, normally as a .conf file.

4.3 Where Data Is Stored

2

Processes

Processes:
Data used by running processes.

Running Processes

Process ID: PID
All processes are given an ID number.

Process Data: /proc/<PID>/
Process data is stored in /proc/ within individual /<PID> folders.

Viewing Running Processes

ps aux : List all processes (BSD format)
ps -eF : List all processes (Linux format)

top : Utility to view running processes and resource usage

Devices

Device Data: /sys
Provides system information regarding attached hardware.

Device Files: /dev
Contains device files (normally block or character devices).

4.3 Where Data Is Stored

3

System Messaging

System Messaging:
Viewing and accessing system messaging.

Kernel Messaging

Kernel Ring Buffer
The kernel ring buffer holds messages related to the operation of the kernel — a ring buffer is simply a buffer of a constant size.

Example

```
$ dmesg | grep sda
[    3.854192] sd 2:0:0:0: [sda] 41943040 512-byte logical
blocks: (21.5 GB/20.0 GiB)
[    3.854235] sd 2:0:0:0: [sda] Write Protect is off
[    3.854236] sd 2:0:0:0: [sda] Mode Sense: 61 00 00 00
[    3.854371] sd 2:0:0:0: [sda] Cache data unavailable
[    3.854373] sd 2:0:0:0: [sda] Assuming drive cache: write
through
[    3.854910]  sda: sda1
[    3.855494] sd 2:0:0:0: [sda] Attached SCSI disk
[    5.320138] EXT4-fs (sda1): mounted filesystem with ordered
data mode. Opts: (null)
[    5.778309] EXT4-fs (sda1): re-mounted. Opts:
errors=remount-ro
```

4.3 Where Data Is Stored

4

Logging

Logging:
Common locations for system and application log data.

Log Daemon

System Logging Protocol: syslog, rsyslogd
Syslog is a service that performs log message collection.

Common Logs

/var/log/messages — General system logs and messages
/var/log/syslog — For Debian-based systems

/var/log/auth.log — Authentication logs
/var/log/secure — For Red Hat-based systems

/var/log/boot.log — System boot logs
/var/log/cron.log — Cron job logs
/var/log/kern.log — Kernel logs
/var/log/faillog — Authentication failure logs

4.4 Your Computer on the Network

1. **Networks and Routers**
   Networks and Routers: The basics of networking, the internet, and routers.

2. **DNS Client Config**
   DNS Client Configuration: The basics of DNS and client configuration.

3. **Network Configuration**
   Querying Network Configuration: How to find and determine the network configuration.

4.4 Your Computer on the Network

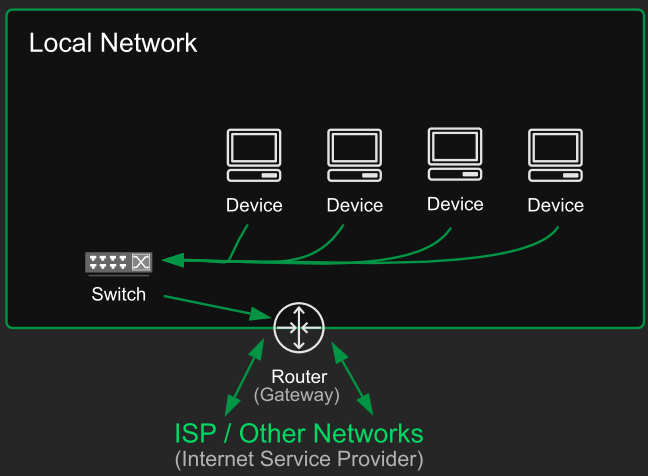
1

Networks and Routers

Networks and Routers:
The basics of networking, the internet, and routers.

Networks

What is a network?
A network is a group of connected devices that are able to communicate with each other.



4.4 Your Computer on the Network

2

DNS Client Config

DNS Client Configuration:
The basics of DNS and client configuration.

What is DNS?

DNS stands for Domain Name System
For devices on the network to be reachable, they must be addressable. DNS maps domain names to their respective IP addresses.

What is the address for www.linuxacademy.com?

The answer is: 52.207.128.102

Example

```
$ dig www.linuxacademy.com
...
;; ANSWER SECTION:
www.linuxacademy.com.  60 IN A 52.207.128.102
www.linuxacademy.com.  60 IN A 52.86.183.13
...
```

Configuration

/etc/resolv.conf
This configuration file is used to determine which hosts to use for DNS queries.

/etc/hosts
Used for statically mapping IP addresses to hostnames.

4.4 Your Computer on the Network

3

Network Configuration

Querying Network Configuration: How to find and determine the network configuration.

Address Configuration

ip route show — Show the current routing table

```
~$ ip route show
default via 10.0.1.1 dev ens33 proto dhcp metric 100
10.0.1.0/24 dev ens33 proto kernel scope link src 10.0.1.197 metric
100
169.254.0.0/16 dev ens33 scope link metric 1000
```

ip addr show — Show the current addresses

```
~$ ip addr show
...
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 ...
    link/ether 00:0c:29:44:93:65 brd ff:ff:ff:ff:ff:ff
    inet 10.0.1.197/24 brd 10.0.1.255 scope global dynamic
noprefixroute ens33
       valid_lft 86385sec preferred_lft 86385sec
    inet6 fe80::d1bc:7e7a:d0c8:8a4e/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

ifconfig — View and change the interface configuration

netstat and **ss** — View listing services and active connections

5.1 Basic Security

1

Root and Standard Users

Root and Standard Users:
Differences between the root account, and user accounts.

2

Systems Users

System Users:
User accounts for servcies and processes.

5.1 Basic Security

1

Root and Standard Users

Root and Standard Users: Differences between the root account, and user accounts.

Standard Users

Standard Unpriviledged Accounts
Standard user accounts are provided a login shell, a home directory, **limited permissions** for view system configurations, and no permission for modiying system configurations. Standard user accounts may be granted the ability to perform priviledged actions using **sudo**.

sudo - super-user do
The use of sudo provides a mechanism whereby a user account is permitted the ability to run a command with elevated permissions.

Root User

The administrative account for the system
The root account has full access to all permissions on the system, and is used for system-level administration tasks. A more secure manner of administration is careful use of **sudo** priviledges.

/etc/passwd | /etc/shadow | /etc/group

5.1 Basic Security

/etc/passwd

This file contains the following seven attribute fields for each local user:

Username - unique login name
Password - empty, moved to /etc/shadow
User ID - unique number ID
Group ID - ID of primary group
GECOS - Long name
Home Directory - Login directory
Login Shell - Log in interpretor

click window to close

5.1 Basic Security

/etc/shadow

This file contains the actual passwords for each local user in an encrypted form. Each line containes eight fields:

Username - unique login name
Password - hashed password value
lastchanged - days since password change
Minimum - Minimum number of days between password changes
Maximum - Maximum number of days between password changes
Warn - Number of days before password expiration to warn user
Inactive - Number of days since expiration
Expire - Absolute expiration date

click window to close

5.1 Basic Security

/etc/group

This file contains group memberships for all local system users, and contains the following four fields:

Group - unique group name
Password - empty
Group ID (GID) - unique group ID number
Group List - a comma deliniated list of usernames that belong to the group

View Group Membership:

```
~$ groups [USERNAME]
```

click window to close

5.1 Basic Security

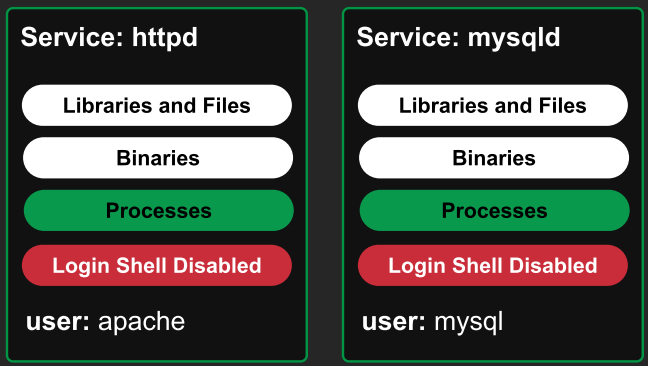
2

System Users

System Users:
User accounts for servcies and processes.

System Users

Application Service Acounts
System users are generally deployed when applications are installed, their home directories are set to application folders, and they normally do not have a login shell. The purpose of having discreet users is to seperate functional priviledges from other applications and services.



5.2 Users and Groups

1

User and Group Commands

User and Group Commands:
Commands for managing users and groups.

2

User IDs

User IDs:
Local users are given a unique ID number.

5.2 Users and Groups

1

User and Group Commands

User and Group Commands:
Commands for managing users and groups.

useradd [options] *username*
Creating new users
The useradd utility is used to add or create users. Options are available for specifying the UID, GID, home directory, and group membership.

/etc/skel
Boilerplate files and folders for new accounts
The contents of /etc/skel are automatically copied when a new user's home directory is created via the useradd utility.

passwd *username*
Update a user's password
The passwd utility may be used to update the current user's password, or another user's with sufficient privileges.

groupadd [options] *groupname*
Creating new groups
The groupadd utility is used to add or create groups.

5.2 Users and Groups

2

User IDs

User IDs:
Local users are given a unique ID number.

User IDs

Numeric ID
All local users are given an identification number that is stored along with the corresponding username in /etc/passwd.

UID 0 — Always the **root** account
UID 1-99 — Traditionally reserved for system users
UID 100+ — Standard users (some distributions start higher)
UID 65534 — Reserved for the user nobody

Remote Users
Typically, a range of UIDs is reserved for non-local users in order to prevent UID collisions with local users.

5.3 File and Directory Permissions and Ownership

1

File and Directory Permissions

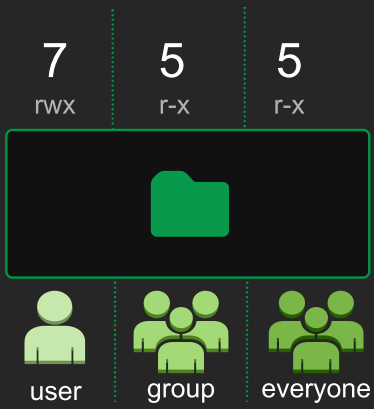
File and Directory Permissions: Permissions and ownership of files and directories.

5.3 File and Directory Permissions and Ownership

1

File and Directory Permissions

File and Directory Permissions: Permissions and ownership of files and directories.



Numeric Permissions
7 — Read, write, execute
6 — Read, write
5 — Read, execute
4 — Read
3 — Execute, write
2 — Write
1 — Execute
0 — No permissions

chown **chmod**

Directory Permissions
Execute = Enter Directory
Directories require execute permissions in order to be entered by the user, group, or everyone.

5.3 File and Directory Permissions and Ownership

```
chown [OPTIONS] USER[:GROUP] FILE(s)
```

Changes the ownership of the file.

Examples

Change a file's user ownership:
$ chown cloud_user file1

Change a file's group ownership:
$ chown :cloud_user file1

Change a file's user and group ownership:
$ chown cloud_user:cloud_user file1

click window to close

5.3 File and Directory Permissions and Ownership

```
chmod [OPTIONS] PERMISSIONS file
```

Changes the permissions of a file.

Examples

Read, write, and execute for all:
$ chmod 777 file1

Read, write, and execute for owner; read for group and everyone:
$ chmod 744 file1

Add execute to current permissions:
$ chmod +x file1

Remove write from current permissions:
$ chmod -w file1

click window to close

5.4 Special Directories and Files

1

Temporary Files and Folders

Temporary Files and Folders: Creating and working with temporary files and folders.

2

Symbolic Links

Symbolic Links: Creating and working with symbolic links.

5.4 Special Directories and Files

1

Temporary Files and Folders

Temporary Files and Folders:
Creating and working with temporary files and folders.

/tmp
Cleared upon system boot
All content in /tmp is **cleared** upon system boot. This directory is used by programs or scripts that require temporary files or directories.

/var/tmp
Cleared every 30 days (depending on distro)
All content in /var/tmp **persists** through a system boot. This directory is used by programs or scripts that require temporary files or directories with more persistence than /tmp offers.

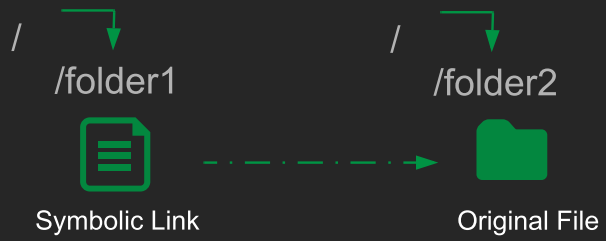
mktemp [OPTION] [NAME TEMPLATE]
Create a temporary file or directory
mktemp can be used to create ad hoc files and directories with a randomized file name portion. These files are not automatically removed.

5.4 Special Directories and Files

2

Symbolic Links

Symbolic Links:
Creating and working with symbolic links.



Symbolic Links

Create a symbolic link to a file or directory

```
~$ ln -s [TARGET] [LINK NAME]
```

The above command creates a symbolic link to the target as a file specified by [LINK NAME].

Example:

```
~$ ln -s /home/cloud_user/file1 /tmp/file2
```