# Linux Basics

- Welcome to week 2; introduction to linux
- Linux architecture
- Linux architecture explained
- Linux distributions
- KALI LINUX ™
- More Linux distributions
- Package managers for installing applications
- Vanilla os
- Nix OS
- Resources for completing Linux labs
- Introduction to the shell
- Different types of shells
- Input and output in the shell
- Linux basics Wrap-up; Glossary terms from week 2

# Welcome to week 2; introduction to linux

Welcome back! We have another important topic to explore.
Previously, you learned aboutoperating systems and user interfaces.
You learned how operating systems work and how resources are allocated in computers.
We also reviewed several common operating systems.
You may already have a favorite operating system.
It's common to hear that people are fans of one over another, but in the security world,
Linux is commonly used.
In this section, you'll be learning more about the Linux operating system and how
it's used in everyday tasks in security.
First, you'll learn about the architecture of Linux.
After this, we'll compare the different distributions of Linux that are available.
Lastly, you'll explore the shell, a key Linux component that allows you to communicate with the system.
I remember when I first learned about the Linux OS, and I'm really happy to explore it with you now.

# Introduction to Linux

You might have seen or heard the name Linux in the past.
But did you know Linux is the most-used operating system in security today?
Let's start by taking a look at Linux and how it's used in security.

Linux is an open-source operating system.
It was created in two parts.
In the early 1990s, two different people were working separately on projects to improve computer engineering.
The first person was Linus Torvalds.
At the time, the UNIX operating system was already in use.
He wanted to improve it and make it open source and accessible to anyone.
What was revolutionary was his introduction of the Linux kernel.
We're going to learn what the kernel does later.

Around the same time, Richard Stallman started working on GNU.
GNU was also an operating system based on UNIX.
Stallman shared Torvalds' goal of creating software that was free and open to anyone.

After working on GNU for a few years, the missing element for the software was a kernel.
Together, Torvalds' and Stallman’s innovations made what is commonly referred to as Linux.

Now that you've learned the history behind Linux, let's take a look at what makes Linux unique.
As mentioned before, Linux is open source, meaning anyone can have access to the operating system and the source code.
Linux and many of the programs that come with Linux are licensed under the terms of the GNU Public License, which allow you to use, share, and modify them freely.
Thanks to Linux's open-source philosophy as well as a strong feature set, an entire community of developers has adopted this operating system.
These developers are able to collaborate on projects and advance computing together.
As a security analyst, you'll discover that Linux is used at different organizations.
More specifically, Linux is used in many security programs.
Another unique feature about Linux is the different distributions, or varieties, that have been developed.
Because of the large community contribution, there are over 600 distributions of Linux.
Later you'll learn more about distributions.

Finally, let's take a look at how you would use Linux in an entry-level security position.
As a security analyst, you'll use many tools and programs in everyday work.
You might examine different types of logs to identify what's going on in the system.
For example,
you might find yourself looking at an error log when investigating an issue.
Another place where you will use Linux is to verify access and authorization in an identity and access management system.
In security, managing access is key in order to ensure a secure system.
We'll take a closer look into access and authorization later.

Finally, as an analyst, you might find yourself working with specific distributions designed for a particular task.
For example, you might use a distribution that has a digital forensic tool to investigate what happened in an event alert.
You might also use a distribution that's for pen testing in offensive security to look for vulnerabilities in the system.
Distributions are created to fit the needs of their users.
I hope you're excited to learn more about Linux.
This will be a very useful skill in the security field.

# Linux architecture

Let me start with a quick question that may seem unrelated to security.
Do you have a favorite building?
And what is it about its architecture that impresses you the most?
The windows? The structure of the walls?
Just like buildings, operating systems also have an architecture and are made up of discrete components that work together to form the whole.
In this video, we're going to look at all the components that together make up Linux.

The components of Linux include the user, applications, the shell, the Filesystem Hierarchy Standard, the kernel, and the hardware.
Don't worry—we'll go into these components one by one together.

First, you are the user.
The user is the person interacting with the computer.
In Linux, you're the first element to the architecture of the operating system.
You're initiating the tasks or commands that the OS is going to execute.
Linux is a multi-user system.
This means that more than one user can use the system's resources at the same time.

The second element of the architecture is the applications within a system.
An application is a program that performs a specific task, such as a word processor or a calculator.
You might hear the word "applications" and "programs" used interchangeably.
As an example, one popular Linux application that we'll learn more about later is Nano.
Nano is a text editor.
This simple application helps you keep notes on the screen.
Linux applications are commonly distributed through package managers.
We'll learn more about this process later.

The next component in the architecture of Linux is the shell.
This is an important element because it is how you will communicate with the system.
The shell is a command line interpreter.
It processes commands and outputs the results.
This might sound familiar.
Previously, we learned about the two types of user interfaces:
the GUI and the CLI.
You can think of the shell as a CLI.

Another element of the architecture of Linux is the Filesystem Hierarchy Standard, or FHS. It's the component of the Linux OS that organizes data.
An easy way for you to think about the FHS is to think about it as a filing cabinet of data.
The FHS is how data is stored in a system.

It's a way to organize data so that it can be found when the data is accessed by the system.

That brings us to the kernel.
The kernel is a component of the Linux OS that manages processes and memory.
The kernel communicates with the hardware to execute the commands sent by the shell.
The kernel uses drivers to enable applications to execute tasks.
The Linux kernel helps ensure that the system allocates resources more efficiently and makes the system work faster.

Finally, the last component of the architecture is the hardware.
Hardware refers to the physical components of a computer.
You can compare this to software applications which can be downloaded into a system.
The hardware in your computer are things like the CPU, mouse, and keyboard.

Congratulations!
We've now covered the architecture of Linux.
An understanding of these components will help you become increasingly familiar with Linux.

# Linux architecture explained

Understanding the Linux architecture is important for a security analyst. When you understand how a system is organized, it makes it easier to understand how it functions. In this reading, you'll learn more about the individual components in the Linux architecture. A request to complete a task starts with the user and then flows through applications, the shell, the Filesystem Hierarchy Standard, the kernel, and the hardware.

## User

The **user** is the person interacting with a computer. They initiate and manage computer tasks. Linux is a multi-user system, which means that multiple users can use the same resources at the same time.

## Applications

An **application** is a program that performs a specific task. There are many different applications on your computer. Some applications typically come pre-installed on your computer, such as calculators or calendars. Other applications might have to be installed, such as some web browsers or email clients. In Linux, you'll often use a package manager to install applications. A **package manager** is a tool that helps users install, manage, and remove packages or applications. A **package** is a piece of software that can be combined with other packages to form an application.

## Shell

The **shell** is the command-line interpreter. Everything entered into the shell is text based. The shell allows users to give commands to the kernel and receive responses from it. You can think of the shell as a translator between you and your computer. The shell translates the commands you enter so that the computer can perform the tasks you want.

# Filesystem Hierarchy Standard (FHS)

The **Filesystem Hierarchy Standard (FHS)** is the component of the Linux OS that organizes data. It specifies the location where data is stored in the operating system.

A **directory** is a file that organizes where other files are stored. Directories are sometimes called “folders,” and they can contain files or other directories. The FHS defines how directories, directory contents, and other storage is organized so the operating system knows where to find specific data.

# Kernel

The **kernel** is the component of the Linux OS that manages processes and memory. It communicates with the applications to route commands. The Linux kernel is unique to the Linux OS and is critical for allocating resources in the system. The kernel controls all major functions of the hardware, which can help get tasks expedited more efficiently.

# Hardware

The **hardware** is the physical components of a computer. You might be familiar with some hardware components, such as hard drives or CPUs. Hardware is categorized as either peripheral or internal.

## Peripheral devices

**Peripheral devices** are hardware components that are attached and controlled by the computer system. They are not core components needed to run the computer system. Peripheral devices can be added or removed freely. Examples of peripheral devices include monitors, printers, the keyboard, and the mouse.

## Internal hardware

**Internal hardware** are the components required to run the computer. Internal hardware includes a main circuit board and all components attached to it. This main circuit board is also called the motherboard. Internal hardware includes the following:

- The **Central Processing Unit (CPU)** is a computer's main processor, which is used to perform general computing tasks on a computer. The CPU executes the instructions provided by programs, which enables these programs to run.
- **Random Access Memory (RAM)** is a hardware component used for short-term memory. It's where data is stored temporarily as you perform tasks on your computer. For example, if you're writing a report on your computer, the data needed for this is stored in RAM. After you've finished writing the report and closed down that program, this data is deleted from RAM. Information in RAM cannot be accessed once the computer has been turned off. The CPU takes the data from RAM to run programs.
- The **hard drive** is a hardware component used for long-term memory. It's where programs and files are stored for the computer to access later. Information on the hard drive can be accessed even after a computer has been turned off and on again. A computer can have multiple hard drives.

# Key takeaways

It's important for security analysts to understand the Linux architecture and how these components are organized. The components of the Linux architecture are the user, applications, shell, Filesystem Hierarchy Standard, kernel, and hardware. Each of these components is important in how Linux functions.

# Linux distributions

Let's learn a little bit more about Linux and what you need to know about this operating system when working as a security analyst.
Linux is a very customizable operating system.
Unlike other operating systems, there are different versions available for you to use.
These different versions of Linux are called distributions.
You might also hear them called distros or flavors of Linux.
It's essential for you to understand the distribution that you're using so you know what tools and apps are available to you.
For example, Debian is a distro that has different tools than the Ubuntu distribution.

Let's use an analogy to describe Linux distributions.
Think of the OS as a vehicle.
First, we'll start with its engine—that would be the kernel. Just as the engine makes a vehicle run, the kernel is the most important component of the Linux OS.
Because the Linux kernel is open source, anyone can take the kernel and modify it to build a new distribution.
This is comparable to a vehicle manufacturer taking an engine and creating different types of vehicles: trucks, cars, vans, convertibles, busses, airplanes, and so on.
These different types of vehicles can be compared to different Linux distributions.
A bus is used to transport lots of people.
A truck is used to transport a large number of goods across vast distances.
An aircraft transports passengers or goods by air.

Just as each vehicle serves its own purpose, different distributions are used for different reasons.
Additionally, vehicles all have different components which distinguish them from each other.
Aircrafts have control panels with buttons and knobs.
Regular cars have four tires, but trucks can have more.
Similarly, different Linux distributions contain different preinstalled programs, user interfaces, and much more.
A lot of this is based on what the Linux user needs, but some distros are also chosen based on preference—the same way a sports car might be chosen as a vehicle.

The advantage of using Linux as an OS is that you can customize it.
Distributions include the Linux kernel, utilities, a package management system, and an installer.
We learned earlier that Linux is open source, and anyone can contribute to adding to the source code.
That is how new distributions are created.
All distros are derived from another distro, but there are a few that are considered parent distributions.
Red Hat® is the parent of CentOS, and Slackware® is the parent of SUSE®.
Both Ubuntu and KALI LINUX™ are derived from Debian.

As we continue, we're going to take a look at some of the distributions most commonly used by security analysts.
The more you understand these distributions, the easier your work will be.

# KALI LINUX ™

In this section, we're going to cover a Linux distribution that's widely used in security and discuss KALI LINUX™.
KALI LINUX™ is a trademark of Offensive Security and is Debian derived.
This open-source distro was made specifically with penetration testing and digital forensics in mind.
There are many tools pre-installed into KALI LINUX™.
It's important to note that KALI LINUX™ should be used on a virtual machine.
This prevents damage to your system in the event its tools are used improperly.
An additional benefit is that using a virtual machine gives you the ability to revert to a previous state.

As security professionals advance in their careers, some specialize in penetration testing.
A penetration test is a simulated attack that helps identify vulnerabilities in systems, networks, websites, applications, and processes.
KALI LINUX™ has numerous tools that are useful during penetration testing.
Let's look at a few examples.

To begin, Metasploit can be used to look for and exploit vulnerabilities on machines.
Burp Suite is another tool that helps to test for weaknesses in web applications.
And finally, John the Ripper is a tool used to guess passwords.
As a security analyst, your work might involve digital forensics.
Digital forensics is the process of collecting and analyzing data to determine what has happened after an attack.
For example, you might take an investigative look at data related to network activity.
KALI LINUX™ is also a useful distribution for security professionals who are involved in digital forensic work.
It has a large number of tools that can be used for this. As one example, tcpdump is a command-line packet analyzer. It's used to capture network traffic.
Another tool commonly used in the security profession is Wireshark.
It has a graphical user interface that can be used to analyze live and captured network traffic.
And as a final example, Autopsy is a forensic tool used to analyze hard drives and smartphones.
These are just a few tools included with KALI LINUX™.
This distribution has many tools used to conduct pen testing and digital forensics.

We've explored how KALI LINUX™ is an important distribution that's widely used in security, but there are other distributions that security professionals use as well.
Next we'll explore a few more distributions.

quick note from the student.. the course says to use it on a VM, however you can use it as your own flavor of Linux for your desktop. its not recommended unless you know what you are doing or willing to wipe the machine :p

# More Linux distributions

Previously, you were introduced to the different distributions of Linux. This included KALI LINUX ™. (KALI LINUX ™ is a trademark of OffSec.) In addition to KALI LINUX ™, there are multiple other Linux distributions that security analysts should be familiar with. In this reading, you'll learn about additional Linux distributions.

# KALI LINUX ™

**KALI LINUX ™** is an open-source distribution of Linux that is widely used in the security industry. This is because KALI LINUX ™, which is Debian-based, is pre-installed with many useful tools for penetration testing and digital forensics. A **penetration test** is a simulated attack that helps identify vulnerabilities in systems, networks, websites, applications, and processes. **Digital forensics** is the practice of collecting and analyzing data to determine what has happened after an attack. These are key activities in the security industry.

However, KALI LINUX ™ is not the only Linux distribution that is used in cybersecurity.

# Ubuntu

**Ubuntu** is an open-source, user-friendly distribution that is widely used in security and other industries. It has both a command-line interface (CLI) and a graphical user interface (GUI). Ubuntu is also Debian-derived and includes common applications by default. Users can also download many more applications from a package manager, including security-focused tools. Because of its wide use, Ubuntu has an especially large number of community resources to support users.

Ubuntu is also widely used for cloud computing. As organizations migrate to cloud servers, cybersecurity work may more regularly involve Ubuntu derivatives.

# Parrot

**Parrot** is an open-source distribution that is commonly used for security. Similar to KALI LINUX ™, Parrot comes with pre-installed tools related to penetration testing and digital forensics. Like both KALI LINUX ™ and Ubuntu, it is based on Debian.

Parrot is also considered to be a user-friendly Linux distribution. This is because it has a GUI that many find easy to navigate. This is in addition to Parrot’s CLI.

# Red Hat® Enterprise Linux®

**Red Hat Enterprise Linux** is a subscription-based distribution of Linux built for enterprise use. Red Hat is not free*, which is a major difference from the previously mentioned distributions. Because it’s built and supported for enterprise use, Red Hat also offers a dedicated support team for customers to call about issues.

with my personal experience, you can use RHEL9 for personal use. also as of July of 2023, its parent company may be trying to break copyright law by limiting access to source code, and close sourcing the project. if they succeed many other distros might die or become less secure or die, like centOS. I likely wont update this but you can google it yourself to see how that went.

# CentOS

**CentOS** is an open-source distribution that is closely related to Red Hat. It uses source code published by Red Hat to provide a similar platform. However, CentOS does not offer the same enterprise support that Red Hat provides and is supported through the community.
p.s. CentOS may be dead because its parent company is trying to kill it at the time of posting this page, google it to see if it has died LOL

# Arch Linux

**Arch Linux** is an open-source distribution known for its simplicity and user-focused design. It adheres to the "Keep It Simple, Stupid" (KISS) principle, offering a minimal base system that users can customize to their needs, reducing potential security risks, provides more control over the system, rolling-release model ensures up-to-date security updates.

A key feature is the Arch User Repository (AUR), a community-driven repository that lets users compile and install packages from source using the Arch package manager, pacman.

# Key takeaways

KALI LINUX ™, Ubuntu, Parrot, Red Hat, and CentOS are all widely used Linux distributions. It’s important for security analysts to be aware of these distributions that they might encounter in their career.

# Package managers for installing applications

Previously, you learned about Linux distributions and that different distributions derive from different sources, such as Debian or Red Hat Enterprise Linux distribution. You were also introduced to package managers, and learned that Linux applications are commonly distributed through package managers. In this reading, you'll apply this knowledge to learn more about package managers.

# Introduction to package managers

A **package** is a piece of software that can be combined with other packages to form an application. Some packages may be large enough to form applications on their own.

Packages contain the files necessary for an application to be installed. These files include dependencies, which are supplemental files used to run an application.

Package managers can help resolve any issues with dependencies and perform other management tasks. A **package manager** is a tool that helps users install, manage, and remove packages or applications. Linux uses multiple package managers.

**Note:** It's important to use the most recent version of a package when possible. The most recent version has the most up-to-date bug fixes and security patches. These help keep your system more secure.

# Types of package managers

Many commonly used Linux distributions are derived from the same parent distribution. For example, KALI LINUX ™, Ubuntu, and Parrot all come from Debian. CentOS comes from Red Hat.

This knowledge is useful when installing applications because certain package managers work with certain distributions. For example, the Red Hat Package Manager (RPM) can be used for Linux distributions derived from Red Hat, and package managers such as dpkg can be used for Linux distributions derived from Debian.

Different package managers typically use different file extensions. For example, Red Hat Package Manager (RPM) has files which use the *.rpm* file extension, such as *Package-Version-Release_Architecture.rpm*. Package managers for Debian-derived Linux distributions, such as dpkg, have files which use the *.deb* file extension, such as *Package_Version-Release_Architecture.deb*.

# Package management tools

In addition to package managers like RPM and dpkg, there are also package management tools that allow you to easily work with packages through the shell. Package management tools are sometimes utilized instead of package managers because they allow users to more easily perform basic tasks, such as installing a new package. Two notable tools are the Advanced Package Tool (APT) and Yellowdog Updater Modified (YUM).

## Advanced Package Tool (APT)

APT is a tool used with Debian-derived distributions. It is run from the command-line interface to manage, search, and install packages.

## Yellowdog Updater Modified (YUM)

YUM is a tool used with Red Hat-derived distributions. It is run from the command-line interface to manage, search, and install packages. YUM works with *.rpm* files.

# Key takeaways

A package is a piece of software that can be combined with other packages to form an application. Packages can be managed using a package manager. There are multiple package managers and package management tools for different Linux distributions. Package management tools allow users to easily work with packages through the shell. Debian-derived Linux distributions use package managers like dpkg as well as package management tools like Advanced Package Tool (APT). Red Hat-derived distributions use the Red Hat Package Manager (RPM) or tools like Yellowdog Updater Modified (YUM).

# Vanilla os

i came accross vanilla os in a yt video and it appears to be realy interesting. i quite like gnome as a base and im thinking this is potentially one of the best options in the future if they stick with it. also theres only two updates per year and a roling release but you have to manually enable rolling releases

tl;dr

its stable asf, and you can run any app on it!

**Vanilla OS: A Unique Solution to Distro Hopping and the Future of Software Installation**

Vanilla OS is a Linux distribution that aims to resolve the common practice of distro hopping. This term refers to the habit among Linux users of switching between different Linux distributions to find the perfect balance of stability, hardware support, and application access. Vanilla OS offers all these features within a single, highly stable base.

What sets Vanilla OS apart is its  approach to software installation. It introduces 'apx', a package manager that allows software installation from any source by installing them onto distro containers. This means that Vanilla OS can run virtually any software developed for Linux, effectively addressing the issue of distro packaging fragmentation. This unique combination of features positions Vanilla OS as a potential game-changer in the Linux ecosystem.

**Intuitive User Experience**

Vanilla OS prioritizes user-friendliness in its design. The intuitive installer guides users through the necessary steps, and post-installation, users can customize their experience, choosing between dark and light mode, enabling support for Flatpak and AppImage, and selecting their preferred apps. This user-centric approach makes Vanilla OS accessible to both beginners and technically inclined users.

**Immutability and Atomicity: Enhancing Security and Stability**

One of the distinguishing features of Vanilla OS is its immutable and atomic nature. The base system is locked down, preventing both applications and users from writing to it, except for certain directories like the home folder or partition and the /etc and /var directories. This design principle significantly bolsters the system's security.

Updates in Vanilla OS are applied atomically, meaning each update either completes successfully or, if any issue arises, the entire operation is reverted, leaving the system unaltered. This ensures that any reboot will either return the system to its previous state or update it successfully.

**Is Vanilla OS the Future of Linux Distributions?**

Vanilla OS represents a promising concept that could potentially shape the future of Linux distributions. It provides access to virtually all Linux-developed software at native speeds within a highly stable base. However, it's not a one-size-fits-all solution. Users who only need software from FlatHub may not require Vanilla OS, and those unfamiliar with the command line or the concept of containers might find it challenging to use. However, with further development, such as a graphical layer on 'apx' for intuitive software installation from containers, Vanilla OS could become an optimal solution for users seeking extensive software access without compromising system stability

for anyone who cares this one was written with ai, modified by me...

# Nix OS

make page on nix os, alternative to vanilla os?

# Resources for completing Linux labs

This course features hands-on lab activities where you'll have the opportunity to practice Linux commands in the terminal. You'll use a platform called Qwiklabs to complete these labs. In this reading, you'll learn how to use Qwiklabs.

This reading first provides a section on how to use Qwiklabs, which includes details on how to launch a lab, how to interact within the Qwiklabs environment, and how to end a lab. This is followed by another section on helpful navigation tips and keyboard shortcuts; these may be useful when working in the terminal.

**Note**: You will not launch Qwiklabs directly from this reading and instead will do this through lab activities and exemplars that you encounter throughout the course. Im not posting the Qwiklabs, this is just how to use Qwiklabs, and if you use Qwiklabs thats on you.

# How to use Qwiklabs

## Launching Qwiklabs

When you select a lab, you start from a Coursera page. You will need to click **Launch App** on that page. After you click **Launch App**, a new tab will open with a Qwiklabs page that contains instructions for that particular lab.

## Start Lab button

On the Qwiklabs page, you must click **Start Lab** to open a temporary terminal. The instructions for the lab will move to the right side of the screen.

Green button with text "Start Lab" and a maximum time limit of 20 minutes displayed.

Read the instructions and complete all the tasks in the lab by entering commands in the terminal.

**Note**: It may take a moment for the terminal to start.

# Lab control dialog box

After you click **Start Lab**, the lab control dialog box opens. It contains the **End Lab** button, the **timer**, and the **Open Linux Console** button.

You can hide or unhide the dialog box by clicking the following icon in the red box:


# The timer

The **timer** starts when the terminal has loaded. The timer keeps track of the amount of time you have left to complete a lab. The timer counts down until it reaches 00:00:00. When it does, your temporary terminal and resources are deleted.

You will have ample time to complete the labs. But, stay focused on completing the tasks to ensure you use your time well.

# Open Linux Console button

When you click the button to **Open Linux Console**, the terminal opens in a new browser window:


Use this feature if you want a full-screen view of the terminal. You can close this window at any time. Closing the window does not end your lab, and you can continue working in the terminal in the original tab.

# Check progress

You can check your progress by clicking **Check my progress** at the end of each task.


If you haven’t yet completed a task, you’ll receive hints on what you must do to complete it.

You can click **Check my progress** whenever you want to check the completion status of a task or receive a hint.

# Using copy/paste commands

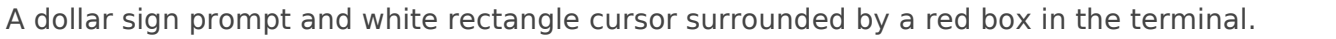
The first time you try to use copy or paste keyboard shortcuts (such as **CTRL + C**), you'll receive a pop-up requesting permission to use your device's clipboard: "**googlecoursera.qwiklabs.com wants to see text and images copied to the clipboard.**" Please click **Allow** if you would like to be able to use these shortcuts in the Qwiklabs platform. If you choose not to allow Qwiklabs access to your clipboard, you cannot use keyboard shortcuts but you can still complete the lab.

# Code block

Certain steps may include a code block. Click the copy button to copy the code provided and then paste it into the terminal.


Two layered rectangles surrounded by a red box on the right side of a sample code block to in

To paste code or other text content that you have copied from the instructions into the terminal, activate the terminal by clicking anywhere inside it. The terminal is active when the cursor in the terminal changes from a static empty outline to a flashing solid block.


A dollar sign prompt and white rectangle cursor surrounded by a red box in the terminal.

Once the terminal is active, use the keyboard shortcut **CTRL + V** (hold down the **CTRL** key and press the **V** key) to insert the copied text into the terminal at the location of the flashing cursor.

# Scrolling

In certain situations, you may want to scroll within the terminal window. To do so, use the scroll wheel on your mouse or the touchpad of your computer.

# End Lab button


A red button with text "End Lab" surrounded by a red box.

Finally, click **End Lab** when you've completed the tasks in the lab.

**Note**: Don't click **End Lab** until you're finished; you'll lose access to the work you've done throughout the lab.

# Tracking progress on Coursera

If you complete a lab but your progress hasn’t been tracked on Coursera, you may need to refresh the page for your progress to be registered. Once you complete the lab and refresh the page, the green check mark should appear.

# Helpful navigation tips and keyboard shortcuts

The following contains a list of navigation tips and keyboard shortcuts you may find useful when completing your Linux labs. Your cursor must be in the terminal window to use these navigation tips and keyboard shortcuts.

- *CTRL + C*: Terminates a command that is currently running; from the instructions portion of Qwiklabs, you can use *CTRL + C* to copy, but within the terminal, it will only terminate a command and if one isn't running, it will display *^C* at the prompt
- *CTRL + V*: Pastes text
- *clear*: Clears the terminal screen; this can also be done by entering *CTRL + L*
- *CTRL + A*: Sets your cursor at the beginning of a command
- *CTRL + E*: Sets your cursor at the end of a command
- *Left arrow* **key**: Moves left within a command
- *Right arrow* **key**: Moves right within a command
- *Up arrow* **key**: Provides the last command you entered into the command line; can be entered multiple times to go through multiple commands from the command history
- *Down arrow* **key**: Provides the next command in the command history; must be after using the *up arrow* key
- *Tab* **key**: Provides available suggestions for completing your text

# Key takeaways

Knowing how to navigate Qwiklabs will be useful as you complete the labs throughout this course. These labs can help you practice what you’ve learned in an interactive environment.

# Introduction to the shell

Welcome back! In this video, we're going to discuss the Linux shell.
This part of the Linux architecture is where the action will happen for you as a security analyst.
We introduced the shell with other components of the Linux OS earlier, but let's take a deeper look at what the shell is and what it does.

The shell is the command-line interpreter.
That means it helps you communicate with the operating system through the command line.
Previously, we discussed a command-line interface.
This is essentially the shell.
The shell provides the command-line interface for you to interact with the OS.
To tell the OS what to do, you enter commands into this interface.
A command is an instruction telling the computer to do something.
The shell communicates with the kernel to execute these commands.

Earlier, we discussed how the operating system helps humans and computers speak with each other.
The shell is the part of the OS that allows you to do this.
Think of this as a very helpful language interpreter between you and your system.
Since you do not speak computer language or binary, you can't directly communicate with your system.
This is where the shell comes in to help you.
Your OS doesn't need the shell for most of its work, but it is an interface between you and what your system can offer.
It allows you to perform math, run tests, and execute applications.
More importantly, it allows you to combine these operations and connect applications to each other to perform complex and automated tasks.

Just as there are many Linux distributions, there are many different types of shells.
We'll primarily focus on the Bash shell in this course.
Let's continue to learn more about the shell.

# Different types of shells

Knowing how to work with Linux shells is an important skill for cybersecurity professionals. Shells can be used for many common tasks. Previously, you were introduced to shells and their functions. This reading will review shells and introduce you to different types, including the one that you'll use in this course.

# Communicate through a shell

As you explored previously, the **shell** is the command-line interpreter. You can think of a shell as a translator between you and the computer system. Shells allow you to give commands to the computer and receive responses from it. When you enter a command into a shell, the shell executes many internal processes to interpret your command, send it to the kernel, and return your results.

# Types of shells

The many different types of Linux shells include the following:

- Bourne-Again Shell (bash)
- C Shell (csh)
- Korn Shell (ksh)
- Enhanced C shell (tcsh)
- Z Shell (zsh)
- PowerShell (by microsoft also on windows)

All Linux shells use common Linux commands, but they can differ in other features. For example, ksh and bash use the dollar sign ($) to indicate where users type in their commands. Other shells, such as zsh, use the percent sign (%) for this purpose.

# Bash

**Bash** is the default shell in most Linux distributions. It’s considered a user-friendly shell. You can use bash for basic Linux commands as well as larger projects.

Bash is also the most popular shell in the cybersecurity profession. You'll use bash throughout this course as you learn and practice Linux commands.

# Key takeaways

Shells are a fundamental part of the Linux operating system. Shells allow you to give commands to the computer and receive responses from it. They can be thought of as a translator between you and your computer system. There are many different types of shells, but the bash shell is the most commonly used shell in the cybersecurity profession. You'll learn how to enter Linux commands through the bash shell later in this course.

# Input and output in the shell

Hello again! In this video, we're going to learn a little more about the shell and how to communicate with it.
Communicating with a computer is like having a conversation with your friend.
One person asks a question and the other person answers with a response.
If you don't know the answer, you can just say you don't know the answer.
When you communicate with the shell, the commands in the shell can take input, give output, or give error messages.

Let's explore standard input, standard output, and error messages in more detail.
Standard input consists of information received by the OS via the command line.
This is like you asking your friend a question during a conversation.
The information is input from your keyboard to the shell.
If the shell can interpret your request, it asks the kernel for the resources it needs to execute the related task.

Let's take a look at this through echo, a Linux command that outputs a specified string of text.
String data is data consisting of an ordered sequence of characters.
In our example, we'll just have it output the string of: hello.
So, as input, we'll type: echo hello into the shell.
Later, when we press enter, we'll get the output.
But before we do that, let's first discuss the concept of output in more detail.

Standard output is the information returned by the OS through the shell.
In the same way that your friend gives an answer to your question, output is a computer's response to the command you input.
Output is what you receive.
Let's pick up where we left off in our example and send the input of: echo hello to the OS by pressing enter.
Immediately, the shell returns the output of: hello.

Finally, standard error contains error messages returned by the OS through the shell.
Just like your friend might indicate that they can't answer a question, the system responds with an error message if they can't respond to your command.
Sometimes this might occur when we misspell a command or the system doesn't know the response to the command.
Other times, it might happen because we don't have the appropriate permissions to perform a command.

We'll explore another example that demonstrates standard error.
Let's input: eco hello into the shell. Notice I intentionally misspelled echo as e-c-o.
When we press enter, an error message appears.

To wrap up, we've covered the basics of communication with the shell.
Communication with the shell can only go in one of three ways: the system receives a command—this is input;
the system responds to the command and produces output;
and finally, the system doesn't know how to respond, resulting in an error.
Later, you'll become much more familiar with this as we explore commands useful for security professionals.

# Linux basics Wrap-up; Glossary terms from week 2

We've made it to the end of this section.
Great work!
Let's recap what you've just completed.

In this section, you learned about the Linux operating system.
We examined the architecture of Linux.
In our exploration of the different distributions of Linux, we discussed some of the most widely used distros in security.
You were introduced to KALI LINUX™, Ubuntu, Parrot, Red Hat, and CentOS distributions.
Finally, you learned about the shell and its role as an interpreter between the user and operating system.

Congratulations! You're doing great, and we have more useful topics to come.
In the next part of the program, you'll learn specific commands to use within the shell while working as a security analyst.
Let's continue on.

---

# Terms and definitions from Course 4, Week 2

**Application:** A program that performs a specific task

**Bash:** The default shell in most Linux distributions

**CentOS:** An open-source distribution that is closely related to Red Hat

**Central Processing Unit (CPU):** A computer’s main processor, which is used to perform general computing tasks on a computer

**Command:** An instruction telling the computer to do something

**Digital forensics:** The practice of collecting and analyzing data to determine what has happened after an attack

**Directory:** A file that organizes where other files are stored

**Distributions:** The different versions of Linux

**File path:** The location of a file or directory

**Filesystem Hierarchy Standard (FHS):** The component of the Linux OS that organizes data

**Graphical user interface (GUI):** A user interface that uses icons on the screen to manage different tasks on the computer

**Hard drive:** A hardware component used for long-term memory

**Hardware**: The physical components of a computer

**Internal hardware:** The components required to run the computer

**Kali Linux ™**: An open-source distribution of Linux that is widely used in the security industry

**Kernel:** The component of the Linux OS that manages processes and memory

**Linux:** An open source operating system

**Package:** A piece of software that can be combined with other packages to form an application

**Package manager:** A tool that helps users install, manage, and remove packages or applications

**Parrot:** An open-source distribution that is commonly used for security

**Penetration test (pen test):** A simulated attack that helps identify vulnerabilities in systems, networks, websites, applications, and processes

**Peripheral devices:** Hardware components that are attached and controlled by the computer system

**Random Access Memory (RAM):** A hardware component used for short-term memory

**Red Hat® Enterprise Linux®** (also referred to simply as Red Hat in this course)**:** A subscription-based distribution of Linux built for enterprise use

**Shell:** The command-line interpreter

**Standard error:** An error message returned by the OS through the shell

**Standard input:** Information received by the OS via the command line

**Standard output:** Information returned by the OS through the shell

**String data:** Data consisting of an ordered sequence of characters

**Ubuntu:** An open-source, user-friendly distribution that is widely used in security and other industries

**User:** The person interacting with a computer