5.1.1 Graphical User Interfaces

Click one of the buttons to take you to that part of the video.

Graphical User Interfaces 0:00-1:11

On a Linux system, you can do almost anything from the shell prompt.

System and network administrators are usually comfortable with the Linux command line interface, and they're not intimidated by long shell commands like this one.

But as Linux's popularity increases, end users are demanding user-friendly environments like the Windows and Mac systems they're used to.

This type of environment is called a graphical user interface, or GUI.

Many Linux distributions offer a graphical user interface that functions a lot like the Microsoft Windows interface. If you use Windows, you're already familiar with most of the components on a Linux GUI. Its basic elements are a screen where dialog windows appear and clickable items.

You'll also find a search field.

Here's a list of desktops you can select.

Here's a list of icons you can use to start applications.

And over here, you can adjust the volume, power the system off, and manage users.

One of the great things about the Linux GUI is that it's very modular.

It's composed of several different pieces that work together to create the GUI. You can mix and match different components together to customize the way it looks and works. This gives you flexibility you just can't get with other operating systems.

The Display Server 1:12-2:03

Now, to fully understand the GUI environment, you have to understand the role of a display server.

In a Linux system, a display server is a program that sits between the graphical interface and the kernel and determines how your programs are displayed. It's responsible for coordinating the input and output of the programs and applications that run the GUI interface. The display server facilitates communication with the rest of the operating system, the hardware, and each program and application that runs the GUI.

It's considered a server because it's capable of displaying, or serving, its output on a remote system that's running the display server software.

The display servers communicate with their clients (or the GUI programs, such as Google Chrome) using a display server protocol.

The two most popular display server protocols are X11, the X Windows System's protocol, and Wayland, the Wayland Compositor's protocol. We'll talk about these in more detail later.

Window Manager - Compositor 2:04-2:53

The next component we need to look at is the window manager, which is sometimes referred to as the compositor.

Even though it's the display server's job to create windows within the GUI environment, it's the window manager's job to customize how they look and behave.

You can choose from a wide selection of window managers for Linux, and each one offers a different look and feel for the GUI environment.

Two common window managers for X11 are Enlightenment and KWin.

Popular managers for Wayland include Sway, Way-Cooler, KWin, and Enlightenment.

So, which choice is best? It depends on what type of users you're designing your system for. Some window managers are elegant and fully featured, and others are very minimal. Generally, I suggest staying away from the minimalist window managers and instead go with the more

advanced window managers, such as Enlightenment, so a wider range of users can comfortably navigate the system.

X11 vs Wayland 2:54-5:28

Now that you understand the window manager's role, let's continue discussing the display server.

Over the years, multiple display server software packages have been implemented on Linux systems. One of the most commonly implemented display servers is based on X Windows, or X11.

One of the earliest implementations of X11 was XFree86. In 2004, the XFree86 Project began distributing new code with a copyright license that the Free Software Foundation considered GPL-incompatible. As a result, the Linux community migrated from XFree86 to the X.Org server.

With X11, the window manager, or the compositor, is a separate piece of software that X11 communicates with.

Here's a simplified version of this process.

After the kernel gets an event from an input device, such as the keyboard, that information is sent to the X11 server.

X11 then determines which window is affected and sends the information to the correct X client program, such as Google Chrome.

The X client looks at the event and decides what to do. For example, the user interface may need some changes.

Since the client doesn't know how to render a change, it sends the required changes back to the X11 server.

X11 must send this information on to the compositor,

which returns this information back to X11 to recomposite the part of the screen where the change is made.

X11 then makes the required change on the display.

In contrast to X11, Wayland considers the display server and the compositor one component, so it can simplify the process for this task.

For example, when the kernel gets the event from the input device, it sends it straight to the Wayland compositor.

The Wayland compositor keeps track of what's on the screens in a scenegraph, a collection of organized nodes. Wayland can search through the scenegraph to determine which Wayland client needs the information.

When the Wayland client receives the event, it updates what the user interface should look like. And, unlike X11, it also renders.

The Wayland client then sends a request to the compositor to indicate the updated region, and the compositor updates the scenegraph information to reflect the change.

Then the compositor re-draws the screen, and the process is finished.

Now, keep in mind that the steps just describe are simplified.

In reality, the Wayland architecture integrates the display server, window manager, and compositor into one process. But you can think of Wayland as a toolkit for creating clients and compositors since it's not a single, specific compositor or window manager. In other words, if you want a different window manager, you can write a new one.

The Role of the Desktop Environment 5:29-6:54

The last thing we need to discuss is the GUI desktop.

The desktop environment leverages the information created by the window manager and then adds a series of tools and utilities to tie all of your GUI components together into one cohesive navigation tool.

Here, we have two examples from the same computer running the same operating system, but with different desktops.

The image on the left was captured when the computer was running the desktop named Cinnamon, while the image on the right was captured on the same computer running the desktop called MATE.

As you can see, the main difference is how options and tools are presented to the end user.

With the Cinnamon desktop, a user can access the application by clicking on the menu icon.

With MATE, you must click the Application menu to access your apps.

Although a fully featured GUI is highly recommended for end user systems, some Linux distributions, such as those designed for security or that only function as a server, either have a very basic GUI or none at all.

As with window manager, many Linux desktop environments are available, and the one you use is, really, your preference.

The nice thing is that a single distribution can often support multiple desktop types, giving users the flexibility to choose the one they like best.

When there are multiple desktops available, the user can often select the Settings icon while logging on to the system and choose whichever they prefer.

GNOME, Unity, Cinnamon, MATE, and KDE are common Linux desktop systems in current use.

Summary 6:55-7:07

That's it for this lesson. In this lesson, we discussed the Linux graphical user interface. First, we talked about the display server's role and the window manager's role.

We talked about the differences between the X11 and Wayland display servers. And then we talked about the desktop environment's role.
