How the Linux Desktop is built

This talk-cum-workshop will be broken down into the following sections:

1. What is a display server?
2. Starting a redundant UI with X.org
3. Drawing windows - Window Manager
4. Toolkits - GTK and QT
5. Desktop Envs
6. Login Managers

So let's jump right into it

What is a Display Server?

A display server is very similar to a web server. Think of a web server containing some files. You access those files through a protocol, typically HTTP. Your web browser takes those files, which are essentially instructions, and renders what those instructions dictate.

The server

A display server here takes the job of the web server. It interacts with the kernel, and send instructions to another program, which we will come to later. The protocol over which these instructions are sent is called the X protocol, and the version currently in use is the 11^{th} version, aka, X11. We will take a look at all of this in detail when we actually do this in the Arch Linux Virtual Machine I've given.

The client

That was some information about the server. Coming to what we do with those instructions, we have an X client running which takes those instructions and draws dots and lines on the screen according to them. It's very basic, and it can't really do anything on it's own.

The result of this separation of client and server

The result of this separation is interesting. It means that you can have an X server running on some machine, and an X client running on another, and if those are connected in any way (one common way is via SSH), then the server can send the client commands and it will draw them on the client's machine! This is commonly referred to as X11 forwarding.

But at the same time, this osome functionality comes at the price of some overhead. If you're not forwarding X11, then the server and client are running on the same machine, and that causes some decreased performance. On top of that, over shoddy network connections, `bitmap-heavy` applications will generally lag a lot.

So this is the overview. You have a server running on the system, sending instructions to fill in dots and lines on the screen, over a protocol called X11, to a client on the system

Starting a redundant UI with X.org

Xorg is the most widely used implimentation of the X windowing system. So now is the time when you all start your Virtual Machines.

If you type in startx, then you'll see some terminals open on screen. We will get to those in a minute, but the point was that to start this whole X windowing system, startx is one commonly used utility. Let's configure it a bit.

There is a file in your home directory called `.xinitrc`. Open it up in nano or vim. Which ever one you like. I'll be doing it in nano, so that people not that familiar can follow along

Let's clear out everything in this file, and write a simple script to start nothing.

Just write `echo "exec twm" > ~/.xinitrc`

Now. What this doing?

This is writing the text `exec twm` into a file called .xinitrc in your home folder. Xorg will look for instructions from this folder and execute them. In this case, we want an interface that's called twm. So we just type in exec twm. When we type in startx, it executes that .xinitrc script, and sets the display variable, finds the geometry of the display, and runs twm!

We just created our first GUI on Linux! Now, let's talk more about this GUI that we just created

Drawing Windows - Window Managers

Window managers are exactly what they sound like. They manage windows. These programs are responsible for drawing the borders of the windows, allocating the screen "real estate" within it's borders to that window and seeing what's in focus.

`twm` is an old, old example of such a window manager. It was developed by this guy called Tom LaStrange (great name) and was initially called Tom's Window Manager, but later renamed to Tab Window Manager, for some reason, when adopted into X.

TWM is an example of a *stacking* window manager, that is, it stacks windows on top of each other. Open it up again and open up a few terminals. You can see that they are on top of each other. That's usually how people are used to window managers. There is another way to place windows on a screen though. In my opinion, this way is far better than the traditional stacking WMs.

These are called tiling window managers, and I have downloading an awesome (get it?) example of it called i3. I haven't installed it, in the interest of showing the whole process of setting it up. You'd wanna first install it. Since this is Arch Linux, you'll be using pacman, instead of i3. I'm sorry for the Arch VM, I just didn't know Debian that well to work with it comfortably enough.

`pacman -S i3`

That'll do the job. Now we wanna tell startx to start i3, not twm. That's easy enough. Just type in `exec i3` instead of `exec twm` and it will start!

Let's open up a terminal on this. Press the windows key, also called the super, meta, or mod key and enter. Open up a few more. You can see that all of these are tiling against each other. The whole idea of this WM is to remove the roll of the mouse, and to save pixels on screen. This way, your applications get the maximum screen real estate. Now you won't just stack windows next to each other. You can change the orientation of the tiling by pressing mod+v for vertical and mod+h for horizontal.

That's about as in depth I'll go into i3. For more details, you can talk to me after this talk or to TK over there (points to TK)

Toolkits - QT and GTK

Now, these window managers have done the job of drawing the windows and handling how they interact. How about the stuff that goes inside the window? The actual interface of the program? Well that's handled by yet another component called a Widget Toolkit. This is the program that offers you widgets like buttons and sliders and menus and styles them too. Some examples of these are QT and GTK.

To show an example of a toolkit, I've downloaded a GTK GUI builder called glade. It's pretty great. To run it, just type startx and we'll run it from there.

You can see here the abundance of widgets that GTK provides. This is what the window manager fills the window real estate with.

The grand finale: The Desktop Environment

For this one, I won't explain anything. Rather, let's just see what's different between a window manager and a full desktop environment. I've downloaded KDE Plasma as well as gnome. I'm going to install GNOME (you can install which ever one you want) and run it to show you guys

`pacman -S gnome`

That should do it, once again.

Running it is as simple as running everything else was. Just replace i3 with gnome-session above, and type in startx.

You can very clearly see what's going on here. You still have a window manager, of course. That's how the windows are being created, resized, place, etc. But now you have system trays, and docks, and workspaces and all the good stuff you know and love. This is the Desktop environment. It brings together all of the components that we've talked about, adds some extra interface elements, and provides you with a complete user experience.

Login Managers

Now, so far we've been typing in startx to start the X windowing system. But usually you are presented with a pleasant login screen that prompts you to enter the password. That is what a login manager is. It is a graphical interface for you to login and it also allows you to switch between Desktop Envs.

If you've installed GNOME like me, then to start gnome's login (or sometimes called a display) manager, just type in

`systemctl enable gdm`

If you installed KDE plasma, then type in

`systemctl enable sddm` (SDDM stands for Simple Desktop Display Manager)

Reboot the system and you should be greeted with a nice login screen.

Conclusion

The whole point of this talk was to clear up some confusion I've been seeing with new users, as well as slightly experienced users. This stuff is hard to wrap your head around if you've come from a windows background. On Windows, the whole interface is one monolithic entity. Here, in the world of Tux, the graphical interface is divided into swappable modules, allowing for greater combustibility and flexibility.