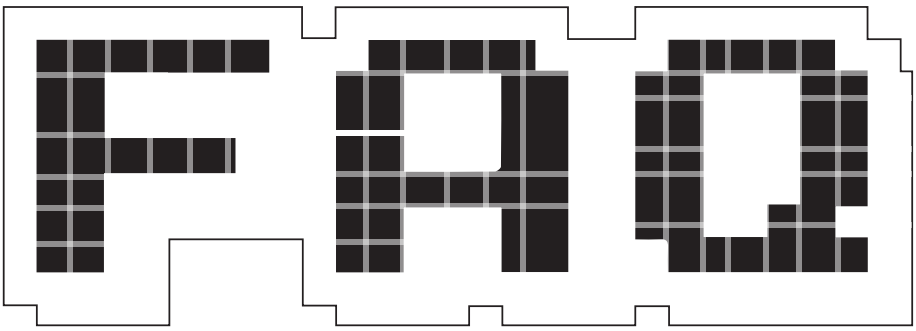


WAYLAND

The Anglo-Saxon god powering your next generation graphical desktop.

GRAHAM MORRISON

Q Why does Wayland need two pages of explanation?

A For most of us, Wayland is difficult to understand because there's nothing tangible to click on. There's no 'About' box to open or configuration panel to play with. But all of these graphical elements can be displayed on your Linux desktop using Wayland – it's just that Wayland hides beneath the surface. The important point is that it's a big improvement over the way this was done before, and is currently done now.

In simple terms, Wayland has the potential to make your desktops talk to your graphics hardware much more efficiently. Developers won't have to work with an arcane system that's massively over-engineered and complicated, while users should see performance benefits and more eye candy. It really could revolutionise the Linux desktop.

Q That sounds promising. What does Wayland actually do?

> **"More often than not it will be toolkits such as GTK or Qt that need to talk to Wayland"**

A Imagine it's the 80s. You're sitting in your bedroom prodding away at a Commodore 64, listening to the Human League and staring through thick-rimmed glasses at your 14-inch colour television. You're coding a game and working on the graphics. To get the best performance, your code is talking directly to the graphics hardware, the venerable VIC-II chip. One of VIC's best features was its ability to allow the programmer to create a simple graphical element, perhaps a spaceship or a gold miner, called a sprite. All the programmer had to do was tell a sprite what to look like, what colour to be and where to appear. They helped the programmer forget about the nuts and bolts of how their computer worked and concentrate on the gameplay.

This is what Wayland helps to do for the modern programmer. They can forget about the nuts and bolts of graphics and concentrate on usability. But because software stacks are now several layers deep, Wayland isn't aimed at the application programmer most analogous to our 80s games programmer. More often than not, it will be toolkits such as Gnome's GTK or KDE's Qt that need to talk to Wayland, and it's these that need to be updated to accommodate its requirements. Application developers shouldn't need to change their code, unless they're using something specific to the way the current system works. As long as the API supports Wayland, the applications will support Wayland and automatically look and feel awesome.

Q Brilliant! So how come it hasn't been adopted already?

A It's not immediately faster than the alternatives, which disappointed many early adopters. Nor is Wayland network-transparent, which means it doesn't include the ability to serve desktop sessions across a network in the way that early X11 did.

The complexity that comes with network transparency is a burden on the current system. That doesn't mean there won't be an alternative, such as a more VNC-like approach to sharing the image buffer to a remote address. It just means that any solution won't be as overbearing. In fact, the Wayland community think that a remote desktop solution using Wayland will be better than VNC on X11 anyway.

Q X11 is the system that Wayland is going to replace?

A Yes. X11 was designed for a different era of computing – the same 1980s of that old Commodore 64. And the key requirement for any 1980s-era enterprise computing installation was for remote graphical terminals. X11 was designed to work across a network so that low powered,

cheap terminals could connect to a centralised computing resource with lots of storage, CPU and RAM.

The apps ran on the central computing resource and sent instructions back on what to be displayed to each terminal. You can still do this with X11 today. In fact, you are doing this today: X11 uses the same client and server configuration even when everything is running on the same machine, making the separation between the client and server a little pointless. There are several other big chunks of X11 that have become redundant, such as its inclusion of some core fonts or big parts of the rendering API – features that are now part of toolkits like GTK and Qt. Then there are the four input subsystems. And that network transparency we were just talking about? It won't work when using modern modern systems with X11 anyway, because of the way they talk to the local graphics hardware.

Q So what's to stop Wayland from being just as bad?

A Apart from the simplicity, there's no legacy code to get in the way of creating a modern graphical subsystem. Just imagine what X11 might be like in another 10 years, and it's difficult to think of how it might adapt to tablets and smartphones.

It's also capable of using hardware specific backends. This won't be necessary for most installations, but there's a backend for the Raspberry Pi that has considerably improved its graphical prowess, when compared to X11, so perhaps performance might improve after all.

Q If it's so hopelessly crufty, how has X11 lasted so long?

A For one simple reason: it works. That's something that can't be said for a great many other technologies. It's stable, despite its complexity, and it's a well understood and a well integrated part of the system. Thanks to the development of many other modules that connect to X11, it's a modern and adaptable solution. But it's never going to get simpler nor better adapted to the kind of computing we do now. All those extensions and plugins, for example, lack any kind of version control, and that means there's no easy way of knowing which features you're going to get when your application supports a different version of a plugin to your X11 installation. Come the revolution, though, you'll still be able to run X11 tools and applications through a compatibility layer called XWayland.

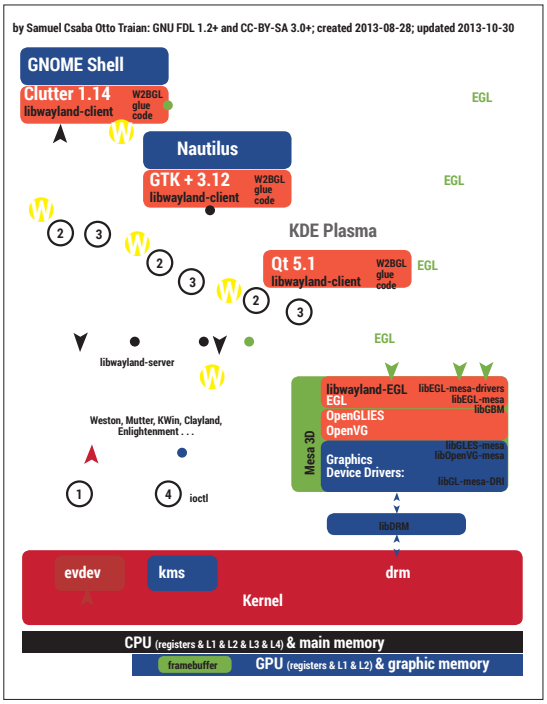
Q Does Wayland do away with the server and client model?

A No. But the client/server model used by Wayland makes more sense. The server is something called the Wayland Compositor, and desktops such as E17 or Gnome, going through their respective APIs, are considered the clients. This is why you always see Wayland described as a 'protocol', rather than a way of rendering graphics. It's the protocol that defines how the clients speak to the server. A server could be replaced with another server, as long as they understood the same protocol. Which is exactly what the people behind Wayland hope will happen. E17 and Gnome both have their own Wayland-compatible compositors.

Q Hang on a mo – what's a compositor?

A You might have first heard the term when desktop effects started to become popular. Compiz, for example, is perhaps the best-known compositing window manager. It adds effects such as wobbly windows, desktop shadows and transparency, and it does this by compositing the contents of the various windows under its control into a single image that can then be used as the desktop. That's why when you run Compiz, you have to replace whatever window manager you're currently using.

In Wayland, the compositor does the same job, only without the help of X11 to turn the final composited image into the desktop you see. It's the server process that pulls all the graphical components together to create what you'd expect to see on the screen. For Wayland, that would mean the server that composes the contents of the various client application windows before sending them on to the rendering stage. These elements already exist, and are not part of Wayland. They're used to get the output from the compositor to your screen.



Wayland is mostly a protocol because it defines how the various components in the stack talk to one another [Image Credit: CC-BY-SA 3.0: ScotXW, based on work by en:Kristian Høgsberg published at en:freedesktop.org: http://wayland.freedesktop.org.

Q If Compiz worked with X11, why is Wayland any better?

A It simplifies the process. X11 was the gateway between the app and the compositing. With Wayland, the applications talk to the compositor without having to go through X11. Wobbly windows with a compositor in X11 worked, but it was much harder if you wanted to tell the compositor you were working with hardware overlays to play back video. Wayland's direct line of communication is a much better way to accomplish the same tasks.

Q Is 2014 going to be the year of Wayland on the desktop?

A We think so, yes. Gnome 3.12 can now operate as a Wayland compositor, bringing native support to the Gnome desktop. So too can Enlightenment's compositor after a huge code dump in the middle of January. The reference compositor for Wayland, named Weston, saw plenty of updates in January's 1.4 release, and there's a new Qt 5.2 based desktop called Hawaii that uses Weston. Even if KDE's Wayland support is slow in making an appearance, you'll definitely be able to migrate to a Wayland-only desktop in the near future.