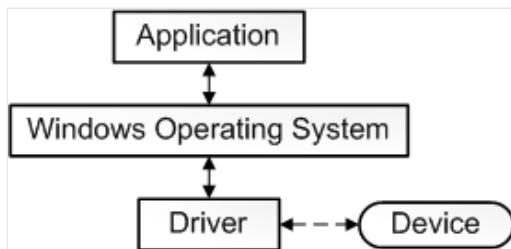


What is a driver?

09/27/2024

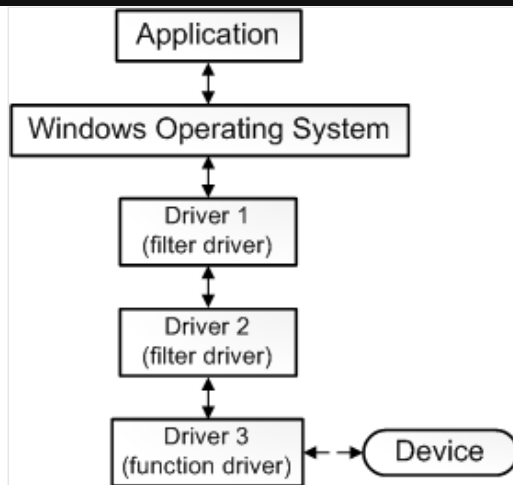
A *driver* is a software component that lets the operating system and a device communicate. For example, when an app needs to read data from a device, it calls a function implemented by the operating system. The operating system then calls a function implemented by the driver. The driver, usually developed by the device's manufacturer, knows how to communicate with the device hardware to get the data. Once the driver gets the data, it gives it back to the operating system, which then gives it back to the app.



Expanding the definition

Our explanation simplifies the concept of drivers. Here are some more points to consider:

- Drivers don't always have to be developed by the device's manufacturer. If a device follows a published hardware standard, Microsoft can write the driver, so the device designer doesn't have to provide one.
- Not all drivers communicate directly with a device. Often, several drivers layered in a [driver stack](#) take part in an I/O request. The conventional way to visualize the stack is with the first participant at the top and the last participant at the bottom, as shown in this diagram. Some drivers in the stack change the request from one format to another. These drivers don't communicate directly with the device. Instead, they change the request and pass it to drivers that are lower in the stack.



Function driver: The driver that communicates directly with the device is called the *function driver*.

Filter driver: Drivers that do auxiliary processing are called *filter drivers*.

For more information on stacks, see [Driver stacks](#).

- Some filter drivers observe and record information about I/O requests but don't actively take part in them. For example, some filter drivers act as verifiers to make sure the other drivers in the stack handle the I/O request correctly.

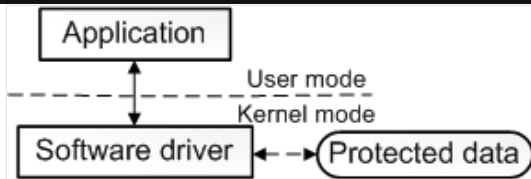
We can refine our definition of a driver as any software component that observes or participates in the communication between the operating system and a device.

Software drivers

Our expanded definition is reasonably accurate but is still incomplete because some drivers aren't associated with any hardware device at all.

For example, if you need to write a tool that accesses core operating system data structures, you can split the tool into two components. The first component runs in user mode and presents the user interface. The second component runs in kernel mode and accesses the core operating system data. The component that runs in user mode is called an application, and the component that runs in kernel mode is called a *software driver*. A software driver isn't associated with a hardware device.

This diagram illustrates a user-mode application communicating with a kernel-mode software driver.



Software drivers always run in kernel mode. They're primarily written to access protected data only available in kernel mode. However, not all device drivers need access to kernel-mode data and resources, so some device drivers run in user mode.

For more information about processor modes, see [User Mode and Kernel Mode](#).

Bus drivers

Another type of driver is the *bus driver*. To understand bus drivers, you need to understand device nodes and the device tree.

For information about device trees, device nodes, and bus drivers, see [Device Nodes and Device Stacks](#).

More information on function drivers

Our explanation so far oversimplifies the definition of *function driver*. We stated that the function driver for a device is the one driver in the stack that communicates directly with the device. This is true for a device that connects directly to the Peripheral Component Interconnect (PCI) bus. The function driver for a PCI device obtains addresses that are mapped to port and memory resources on the device. The function driver communicates directly with the device by writing to those addresses.

However, in many cases, a device doesn't connect directly to the PCI bus. Instead, the device connects to a host bus adapter that is connected to the PCI bus. For example, a USB toaster connects to a host bus adapter (called a USB host controller), which is connected to the PCI bus. The USB toaster has a function driver, and the USB host controller also has a function driver. The function driver for the toaster communicates indirectly with the toaster by sending a request to the function driver for the USB host controller. The function driver for the USB host controller then communicates directly with the USB host controller hardware, which communicates with the toaster.

