**Operating Systems and Architecture**

So far we’ve looked at individual elements of a computer system – the CPU, memory, peripherals, buses, network connectivity and low-level software systems.

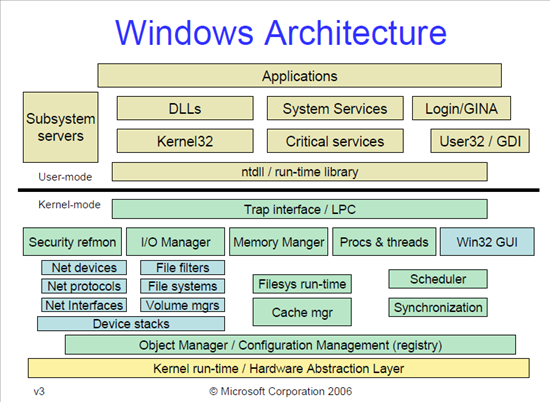
However, computers are useful only if these elements are brought together and made accessible via an **operating system**. There are many different operating systems and they have different components.

**Embedded** operating systems – also called ‘headless’ operating systems – are designed to operate appliances and embedded computers in a variety of applications. For example, the controller in your washing machine; the software in Alexa; or the flight avionics in a modern airplane.

**Windows** operating system – This is a proprietary operating system by Microsoft, who have been active in this field for over 40 years. This is a common OS that you will be familiar with already – applications are launched via program shortcuts, which launch executable programs. Alternatively, programs run as services underneath the OS.

**\*nix operating system** – These are also extremely common operating systems within organisations and enterprises. They can often run graphical user interfaces for the benefit of users, but this isn’t necessary. This type of OS comes in distributions, which are varieties of the OS, and are often (but not always) open-source.

**Windows OS Architecture**

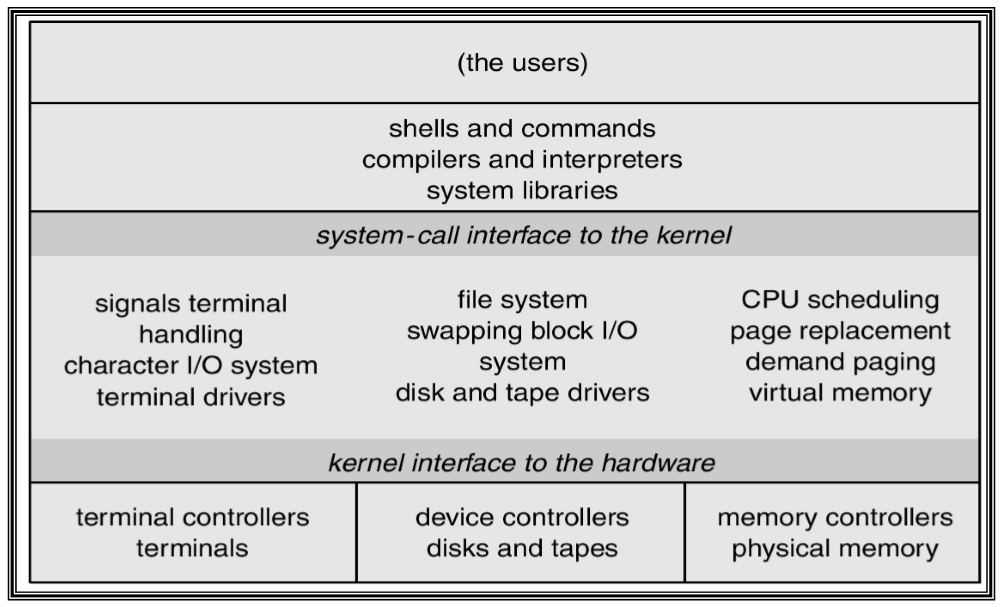
[](https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwjE8oK2lpHhAhWKnxQKHfDSDAEQjRx6BAgBEAU&url=https%3A%2F%2Fresources.infosecinstitute.com%2Fwindows-architecture-and-userkernel-mode%2F&psig=AOvVaw1SqOy5UK8oQkIhRaFYjwVb&ust=1553186949884027)

Windows architecture is split into two key ‘modes’ – the user mode, and the kernel mode. The kernel mode is the mode that runs all the background processes – these are programs that enable the OS to continue functioning and interact with the hardware. For example, the cache manager stores and retrieves pages of data from memory, making sure we don’t need to read from disk for every operation we do. A network interface will send and receive information via a network interface card (NIC) according to the protocols that the interface supports (like TCP/IP).

The user mode, however, is the interactive portion of the OS. The user mode enables us to use the computer – to launch applications, and to use programs that run as a service. These services in the user mode can also be background processes, but they are not intrinsic to the successful operation of the OS – for example, an antivirus program, or a process that keeps your browser updated to the latest version are both user-mode processes.

An important component in the kernel mode is the HAL – the Hardware Abstraction Layer. This is the layer that you can see expressed via Device Manager. This layer is responsible for the low-level hardware calls to and from the physical hardware in your computer – to, for example, write some data to disk. This layer translates the instruction to write to disk from the application to a signal which the hardware can understand.

**\*nix OS Architecture**



The UNIX and Linux architectures (together referred to as \*nix) are substantially different from the Windows ones. These OSes also have a kernel, as does Windows, and the kernel is responsible for communicating with the hardware and running the basic processes that allow the OS to function. The HAL is defined as a series of controllers. Each controller is responsible for communicating with some piece of hardware – for example, the memory controller is able to send the signals to memory to store and retrieve pages from memory.

In these operating systems the user can still run programs, but there is more parity between the system-level background processes and the user ones. There is also more modularity – users can install packages, and these packages can depend on other packages (or provide functionality to other packages). These OSes tend to be more configurable and reliable than larger and more complex Windows-based OSes which makes them ideal for enterprise applications.

**Glossary**

|  |  |
| --- | --- |
| **Term** | **Definition** |
| User interface | The means by which a user interacts with the operating system. This can be graphical (GUI), via the command line (Command Line Interface, or CLI), or even through more esoteric means – voice, for Alexa and related devices, and touchscreens. |
| Operating system | The collection of software that abstracts the hardware in a computer system to a usable, universal software interface to enable a user to gain value from it. |
| Kernel | The interior of an operating system – the core. Normally incorporates the underlying programs, drivers, HAL and/or controllers. |
| Driver/Controller | Software responsible for communicating with a hardware component or peripheral. For example, a driver than can communicate with a printer. |
| Library | Collection of methods that can be used by a program during run-time. On Windows systems these are often labelled .dll. |
| Peripheral | Some hardware component that is plugged into the main computer system, such as a printer, speakers or a monitor. Requires compatible drivers. |
| Shell | A CLI (command-line interface) session that allows the user to use the operating system of the computer it is connected to. Not necessarily the computer from which the user is working. |

**Demo**

Now we will look at what Windows does behind the scenes…

* Task manager
* Resource monitor
* Performance monitor
* Registry editor
* System directories
* Background processes
* Boot processes
* System configuration

**Video**

Differences between Windows and Linux (16:15)

<https://www.youtube.com/watch?v=O3n6bArDEbc>