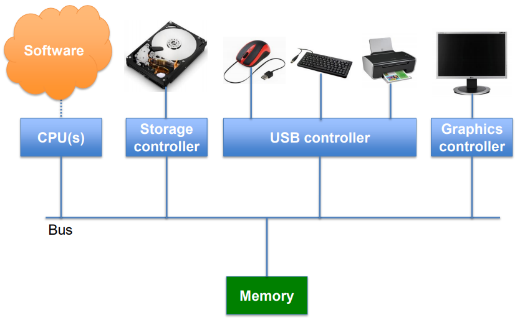
Operating Systems

Introduction

Modern Computer Systems

Modern computer systems now consist of many different parts that all have to work together in order to carry out computations or perform desired tasks. The rise of personal computing has generated the need for a generic structure that allows all of these parts to be easily repurposed, this is where the operating system comes in.

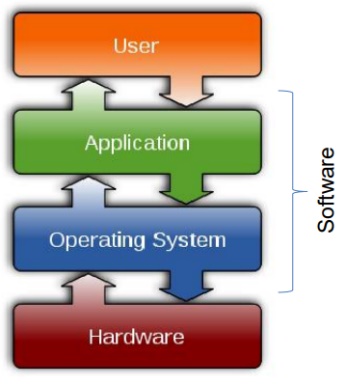
What is an Operating System

An operating system is a program that manages the computer’s hardware. The operating system acts as an intermediary between the user of a computer or the applications on a computer and the computer hardware.

There are many examples of operating systems including Windows, Linux, android, iOS, MS-DOS etc…

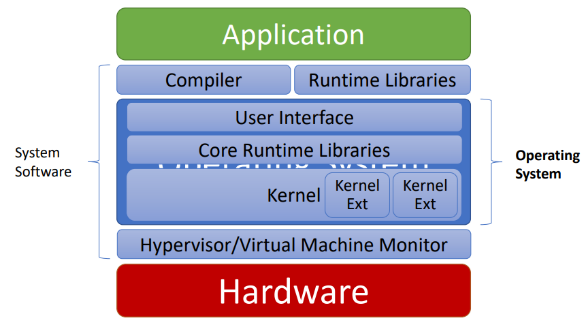
Goals of Operating Systems

The goal of the operating systems include:

* to simplify the execution of user programs and make solving problems easier
* to use the computer’s hardware efficiently, allowing sharing of hardware and software resources
* to make application software portable and versatile
* to provide isolation, security and protection among user programs
* to improve the overall system reliability, error confinement, fault tolerance and reconfiguration

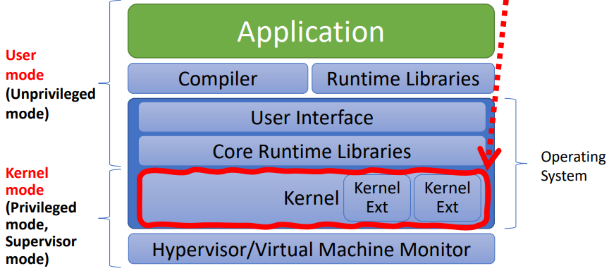
The Traditional Picture

The operating system can be seen as everything you don’t need to write in order to run your application. The OS can be thought of as a library, all operations on I/O devices require OS calls (syscalls), though this isn’t a perfect analogy, you use the cpu/memory without OS calls and it intervenes without having been explicitly called.

What is an Operating System

This image goes into more depth in regards to what is and isn’t part of the operating system. As you can see, the operating system is part of a group of software called the system software and is comprised of several key components.

We can also introduce the idea of User mode and Kernel mode:

The kernel mode gives parts of the OS permission to execute otherwise restricted functions.

This course shall specifically focus on the Kernel (highlighted in red)

The OS and Hardware

It is important to understand the relationship between an operating system and the hardware.

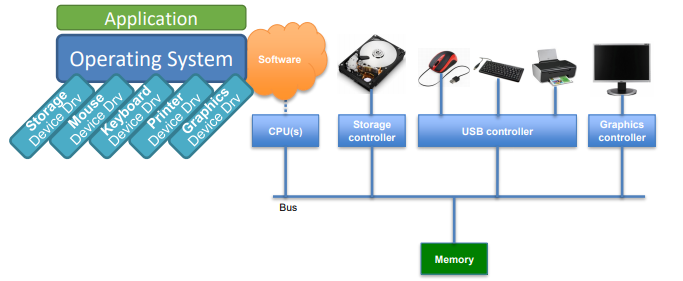
An OS mediates programs’ access to hardware resources (sharing and protection) such as computation (CPU) volatile storage (memory) and persistent storage (disk), network communications (TCP/IP stacks, Ethernet cards, etc…) and I/O devices.

The OS also abstracts hardware into logical resources and well defined interfaces to those resources to make them easier to use, this is done using processes (CPU, memory), files (disk) and sockets (network).

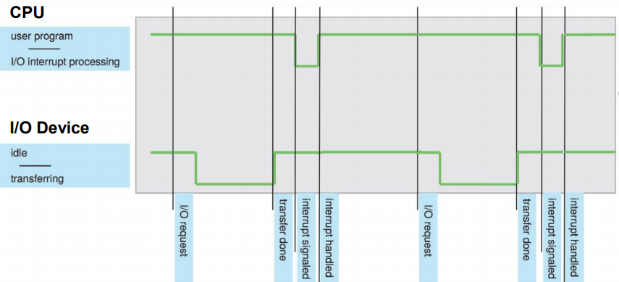
Why Bother With an OS?

There are many benefits of using an operating system for both applications and users. The application benefits include programming simplicity, high-level abstractions are used (like files) instead of the low-level hardware details (such as device registers), these abstractions are reusable across many programs. The OS also makes portability possible (across machine configurations and architectures), as long as the operating system works on that architecture, the programs will all be happy. User benefits include improved safety, each program sees its own virtual machine and thinks it owns the hole computer while only actually having access to a very small part, the OS protects programs from each other and fairly multiplexes resources across programs. The OS also improves efficiency (cost and speed) making it possible to share one computer across many users and concurrently execute multiple programs.

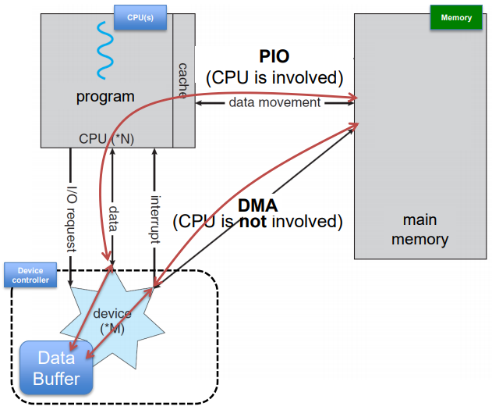
Hardware Recap: Devices

The devices in a computer system are for interacting with the external world (e.g. with users). Every device has a device controller which runs in parallel to the CPU, has buffers for data (local memory) and may move data to main memory like the CPU.

Operating systems have devices drivers for each device controller, in the image, the device controllers are the blue boxes and the devices drivers and the angled boxes.

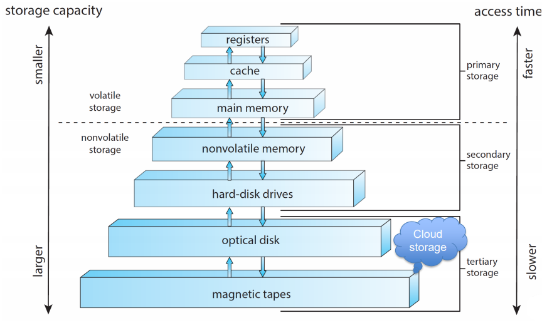
Hardware Recap: Interrupts

Interrupts are very important signals from hardware that alert the system that an event has occurred. Often these events are I/O, such as key press or disk transfer completion. You can see from the graph a user program makes an I/O request, the I/O device then moves from idle to transfer the requested data to memory (for example), when this is done an interrupt is signalled causing the CPU to stop running the user program to handle the new data and then return the control to the program.

Hardware Recap: DMA

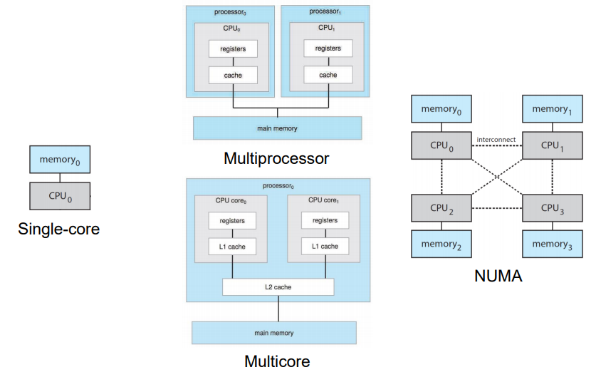
Take an example of a device that can access the disk, this device can take data from the disk and store it in a buffer in its device controller (this could also be network card receiving package etc...). The device can then send an interrupt and the cpu can take this data and move it into the main memory, this is called PIO. The problem with this is the CPU has to ‘manually’ tansfer the data to memory, halting any program in the progress. DMA (Direct Memroy Access) is a solution to this problem, instead of having the cpu move the data, once an interrupt is received, the device driver will program a DMA controller (maybe on the device itself) telling it to move the contents of the data buffer directly to memory, the cpu doesn’t have to move the bytes itself. Once the DMA transfer is done, another interrupt will be sent to inform the CPU the data is ready.

Hardware Recap: Storage Structure

Looking at the image to the right, as we move down in storage types, the type of memory becomes slower but also cheaper. Because of this the higher up the type of memory the closer to the CPU and application running it is used (and less there is in the system).

There are also two types of storage, volatile and non-volatile, volatile storage will loose its contents when it looses power (and non-volatile will not) and as such is generally not used for disk storage (though there are odd exceptions).

Hardware Recap: Memory and CPU

Originally computers were a single CPU, connected to one main memory. But more modern computers use multiple CPUs that all work together and share the same main memory, these systems have become standard.

The last configuration in the image is the NUMA configuration, it is a combination of multiprocessor and multicore. NUMA stands for Non Uniform Memory Access, we may have multiple multi-core chips, connected though high speed connections, in it CPU­­0 can easily and quickly access memory0 but to access memory3 it would have to go through either CPU3 or CPU2 and CPU3.