An Operating Systems's Functions

Alex Hirsch
alexander.hirsch@student.uibk.ac.at

2013-05-30

This document will give a *very* basic overview about the functionality an operating system needs to have. The beginning will provide a general overview describing the major parts. Next will follow a more detailed explanation of the two most important functions (hardware abstraction and process management). Since this paper has to be quite short, the text focuses on simplicity and will state references to more complex yet more educational readings.

1 Basics

The most basic function an operating system (OS) should provide is the ability to abstract the hardware in a way a user can interact with it - without needing to know what is happening inside of the computer. Furthermore, a computer should be capable of running multiple programs simultaneously and since resources (time, memory, etc.) needed by these programs are limited, there has to be some kind of supervisor which manages these program's execution.

A common feature of modern computer programs is "inter process communication". Computers can usually run multiple processes and the OS has to manage these processes as well and provide a way for them to interact / communicate with each other.

Last but not least the OS has to provide an API for programmers to create applications utilizing the hardware underneath, as well as manipulating the resources to fit their needs.

A more advanced description can be found at [3, p. 49]

Figure 1 shows a very basic system consisting of basic computer parts and a user. The user interacts with the OS and with applications in order to achieve these tasks. He can't access the hardware directly, all hardware related instructions are done by the OS. This should bring the significance of providing a good API for hardware access to mind.

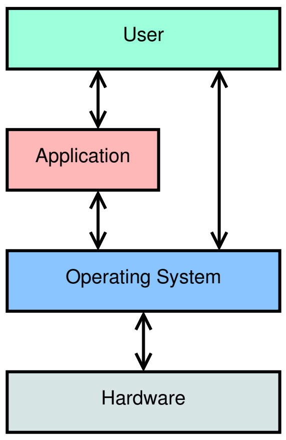

Figure 1: a very basic computer model

2 Hardware Abstraction

Abstraction is an always present topic in computer science starting from the very bottom where the flow of electrons are regulated using electronic circuits up to the point where a single button can do almost everything. Here the operating system plays a very crucial part, it abstracts the hardware underneath in a way which makes it easy for programmers as well as for users to use it.

The amount of abstraction can be very different. For example, the usage of a RS232 serial port is not abstracted very much. The operating system just takes away the data packaging (preamble, Baud, parity). Reading and Writing data is left for a programmer to implement. Each byte has to be send and received by hand.

On the contrary side when accessing a simple text file, a whole lot of things happen. You do not need to provide information about cylinders or sectors when accessing a hard drive. Your operating system has already abstracted the underlying hardware and created a special data structure for files, commonly called a *file system*. See [2] and [1, p. 481] for more information.

Without the abstraction an OS creates for you, it would be sheer impossible to create advanced computer programs, especially when utilizing special hardware is needed.

3 Process Management

As already mentioned above, another issue modern computers have to face is concurrency. There are a lot of things happening and a computer needs to react on all of them.

Even if you are only using one application at a time, your computer uses a whole bunch of different programs to manage the system. To keep things organized processes have been invented. A process represents the execution of a single program with all needed resources (memory, file handles, etc.) and background information. [1, p. 56]

The current generation of computers run multiple times more processes than CPU cores are present. Which leads to the issue that not all processes can be executed at the same time. Time multiplexing is the solution we encounter here. Each process is granted a certain amount of time by the OS. In this *time slice* the program related to the process is executed. Is his time limit reached, the process will be suspended (its current state is stored) and another suspended process will be continued (state will be recovered).

Suspending and resuming multiple processes is happening so fast it looks like they run in parallel. And everything is managed by the operating system.

References

[1] Albert S. Woodhull Andrew S.Tanenbaum. *Operating Systems Design and Implementation*. Pearson Education, 3rd edition, 2006.

[2] Holger Kreissl. Grundlagen der informatik - betriebssysteme. http://www.kreissl.info/bs_inhalt.php, 2004.

[3] William Stallings. *Operating Systems Internals and Design Principles*. Prentice Hall, 7th edition, 2012.