OPERATING SYSTEM

Miroslaw Malek,
Institut fuer Informatik, Humboldt-Universitaet zu Berlin, Berlin, Germany

Keywords: Batch Processing, Compiler, Computer Architecture, Computer Organization, Concurrency, Consistency, Deadlock, Executive, File, File System, Input/Output System, Job (Process, Task) Scheduling, Kernel, Linker, Loader, Memory Hierarchy, Memory Management, Multiprocessing, Multiprogramming, Multitasking, Mutual Exclusion, Priority, Process, Protection, Security, Scheduling, Task, Virtual Memory

Contents

Summary

Operating systems are integral part of computer software whose goals are ease and efficiency of use of computer systems. After a brief introduction and history of operating systems, the views, types and services of operating systems are described and the basic modes of operation including program execution and input/output processing are given. Next, the file system basics, control, protection and allocation are presented followed by job and resources scheduling methods. Also, some other problems such as concurrency, consistency, deadlocks, and error control are described. Finally, distributed operating systems concepts, web computing and the outlook for the future are outlined as well.

1. Introduction

Operating system (OS) is a program or rather more frequently a set of programs whose primary purpose is to provide an interface between applications programs such as word processing or table calculations and a computer hardware such as processor, disk or keyboard (see Fig. 1). The objective is to build a machine that is a pleasure to work with but at the same time ensure ease of use, efficiency, good performance and high utilization of available resources which include hardware such as the central processing unit (CPU), memory unit and I/O devices as depicted in Fig. 2 and software as shown in Fig. 3.

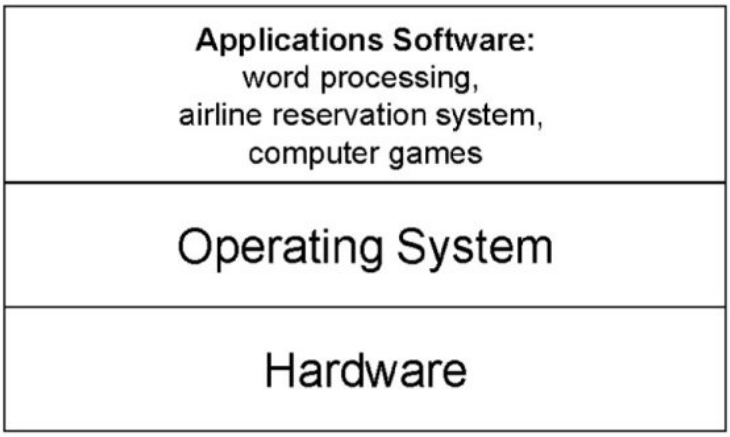

Figure 1: Operating system as an interface layer between computer hardware and application program.

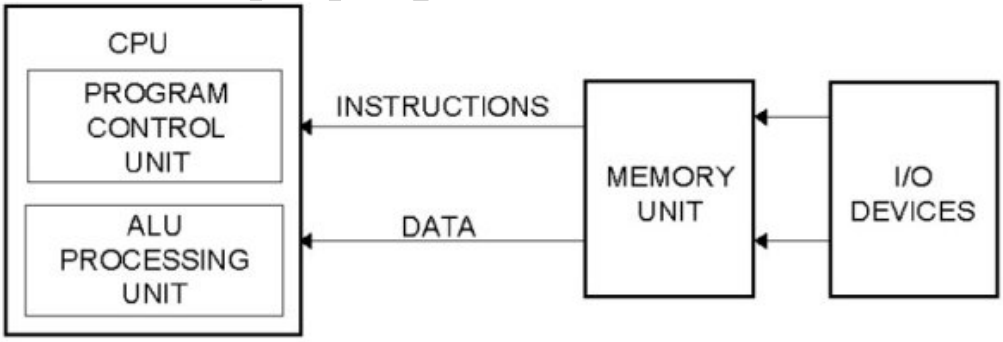

Figure 2: Computer organization (hardware resources).

More formally, the OS performs various supervisory and control functions over the users programs in execution which are called tasks or processes. The OS is responsible for:

- Creating and killing processes
- Overseeing and controlling the progress of processes executions (e.g., a detection of an infinite loop should lead to a timeout or a transfer of control to another program)

- Scheduling, i.e., defining the order in which processes are to be executed
- Handling of exceptions, errors or protection violations
- Allocating hardware and software resources to processes
- Providing access to software and data, e.g., compilers and assemblers, library programs, text files, pictures, music
- Ensuring required quality-of-service with respect to security, real-time, dependability and others
- Supporting communication and web services (even a browser is viewed by some as a part of the OS).

Computer systems belong to a class of the most complex and complicated devices ever conceived by humanity. The integral part of the computer system is the OS. The size of the OS may range from a few hundred lines of code for a very primitive one to 50 million lines as in the case of Windows/NT. It took eight years and thousands of people to develop such a system and as it is known the result is not perfect as the sheer complexity of this undertaking makes reaching the perfection impossible.

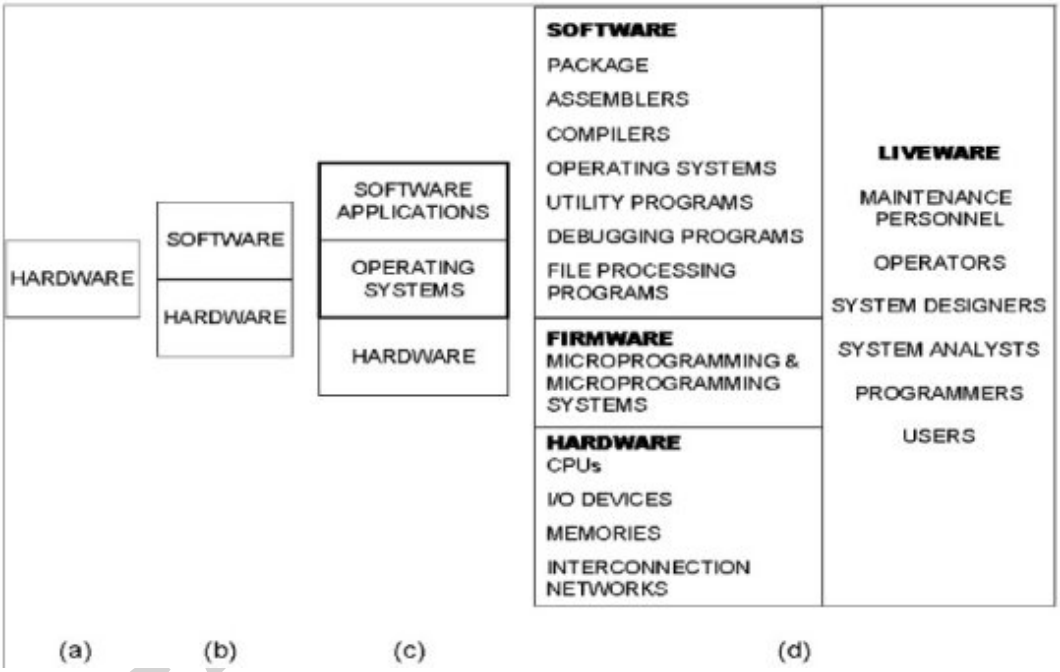

Figure 3: Operating systems evolution_

What follows is: first, a brief history of operating systems (see also History of Computation), various views and types of OS'es and services provided by them, followed by the description of essential inner workings of the OS and its major parts such as file system and memory management as well as the resource allocation and scheduling. Some other problems such as concurrency and consistency control, deadlocks, and error handling are presented as well.

Finally, a brief description of the distributed operating systems, web computing and the outlook for the future are given.

2. Brief History

Initially, computers were conceived as bare mechanical, electrical or electronic devices (hardware) where the user manipulated and controlled the computation.
It was not until the 19th century when Charles Babbage, a mathematics professor at Cambridge University, introduced a concept of programming in his mechanical computers, the Difference Engine (1822-32) and the Analytical Engine (1833-71) which were able to automatically solve a number of complicated problems such as a function approximation or finding roots of a set of equations. Later, Alan Turing (1936) introduced an abstract concept of the universal computing machine which was capable of processing a sequence of instructions (a program) and solve any data processing problem. In the fourties, John von Neumann (born in Hungary, taught at the University of Berlin, known today as Humboldt University, and later Princeton University) suggested a concept of conditional control transfer from one program to another that became essential in the OS design.

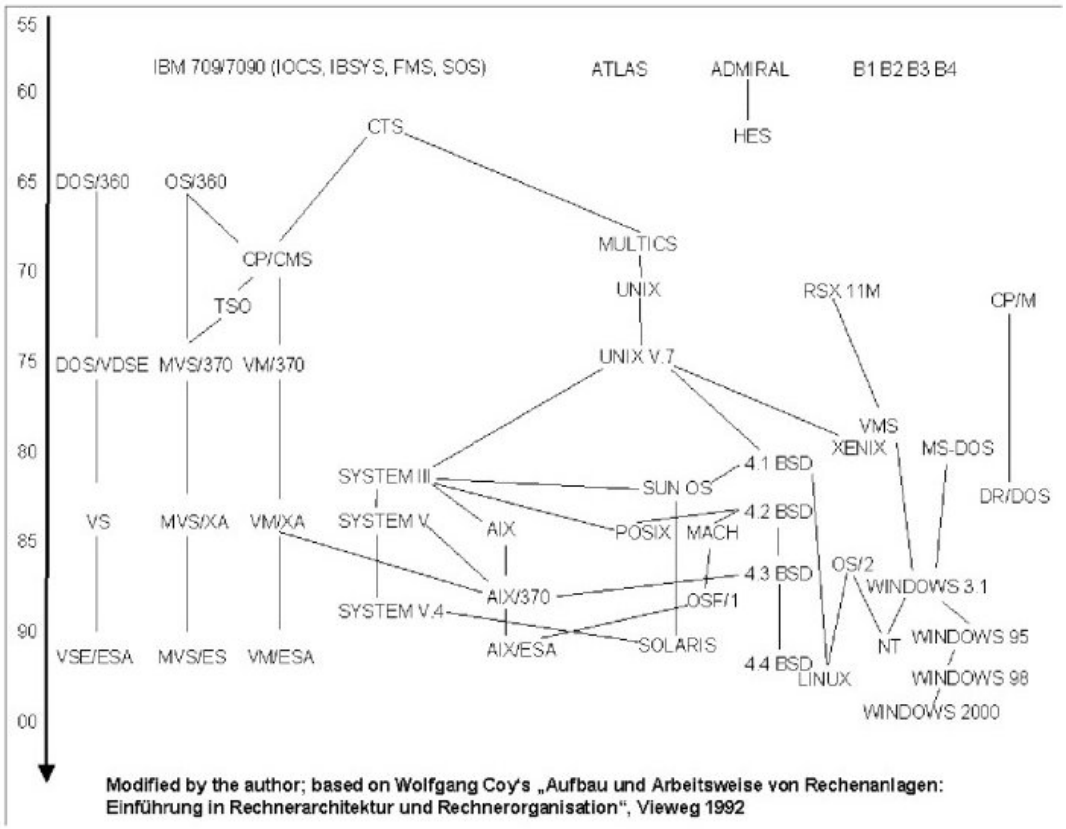

Figure 4: Evolution of operating systems.

In the late fifties, the first rather simple OS'es have emerged such as IBM's Input Output Control System (IOCS), IBSYS, Fortran Monitor System (FMS), SHARE OS (SOS) and Manchester University's Atlas I Supervisor which introduced system calls and virtual memory (Fig. 4). The evolution progressed and the computer designers realized that in addition to computer hardware, computer software plays vital role as

well (Fig. 2b). With growing number of applications and advances in the OS design a clear distinction between the user programs, called applications, and the software responsible for the proper functioning of a machine, called operating system, has emerged (Fig. 2c). Over the years, firmware, a set of programs for direct execution of machine instructions, called microprograms has been added to some machines and the users, called by some liveware, have been divided into several specialized groups such as computer designers, system analysts, programmers, operators, maintenance personnel, and customers or users in general sense (Fig. 2d). Lately, one more term, middleware, has emerged and depicts, in general, all software that is not the OS nor applications programs such as communications software (mainly protocols), diagnostic programs and others.

Early operating systems were custom-designed by each computer manufacturer to run on their hardware. The competition was in features of both the operating system and the underlying computer hardware.

The features have been evolving from very rudimentary ones such as the ability of automatic loading of programs and data (these processes of the OS would be called loaders) to linking various programs into one application by so called linkers.

With progress in machine design, assemblers were added, i.e., programs translating the specific for a given computer assembly language code (machine instructions) into the machine language of 0's and 1's.

With emergence of high level languages such as FORTRAN further progress was made to ease the programming tasks and another type of translator program, called compiler, was added to automate the translation of a high-level language source code into the machine level object code.

To optimize utilization of the programs with similar resource needs the programs would be grouped into batches and that way the so called batch processing was born.

Loaders, linkers, assemblers, computers and job sequencing software would form a rudimentary operating system, often called resident monitor, monitor, supervisor or executive (later also a kernel).

With the advances of the third generation computing the IBM 360 mainframe (a large computer) has established new standards for hardware and software design. Buffering and spooling were used to prepare jobs residing in I/O for more efficient processing or conversely for outputting.

The next challenges were portability (the ability to run a given operating on different computers), and compatibility (the ability to run a given operating system on several generations of a given machine, e.g., IBM 360 and IBM 370 series). Then the whole new generation of OS'es has been developed based on the MIT's operating system, called Multics. Especially, for minicomputers such as the most popular one in its days, the PDP-11, two new generation operating systems have been developed, Digital's RSX-11M by David N. Cutler and AT&T's Unix by Thompson and Ritchie who

developed Unix by accident rather than a commercial necessity. The main challenge was to pack the OS into 32 Kbytes of memory as the other 32 Kbytes have been reserved for applications. The overlays had to be used enabling swapping in and out various parts of the OS on demand. The systems supported multiprogramming, hierarchical file system, application swapping and real-time scheduling. The next Digital's operating system, called, VMS, was able to execute mainframe applications on a minicomputer. It was compatible with Digital's earlier computers and supported a concept of virtual memory. Time sharing features were added to further improve the CPU utilization and then the interactive computing was introduced. Linus, Torvalds, a Finnish student, developed an ultra small Unix, called Linux, that could easily fit on a personal computer. Linux has a modular design ensuring high dependability and portability. Time-shared systems allowed multiple users simultaneous access to a given computer, and then the advances in communication lead to connecting many computers together which in turn gave birth to distributed computing and in turn distributed operating systems.

In 1988 David Cutler moved to Microsoft and developed with his team the next generation of the operating system, called Windows, to support portability, multiprogramming, distributed computing, scalability, standards compliance and globalization (various language options could easily be installed and updated). A pretty complete chronological development of operating systems is shown in Fig.4.

Also, a special-purpose operating systems, especially for telecommunication and automobile industry began to flourish and the challenges focus mainly on ubiquity, platform independence, mobility, high performance, dependability and security.

3. Types of Operating Systems

Depending on the type of computer systems and applications, different types of operating systems have been developed. Operating systems can be categorized according to their functionality. There are operating systems for supercomputers, distributed computers, mainframes, servers, workstations, personal computers, real-time systems, fault-tolerant systems, graphics engines, palm computers, cellular phones and the entire array of embedded systems, such as automobile control systems or a communication server.

The OS'es for common use on mainframes, workstations and personal computers such as Unix and Windows are frequently referred to as general-purpose operating systems while the others are considered as special-purpose OS'es.

The OS can be open or proprietary. Open, such as UNIX operating system, means that the source code is available to anybody who wants it and the users/programs can easily extend and add the additional features to the system. Closed or proprietary such as Windows operating systems means that the code is not accessible to the users and they have to accept the OS as delivered by manufacturer.

The OS'es can also be classified based on the number of programs, users and machines they can handle. The simplest OS is for a single program, single user, and single

processor, the most complex one can handle multiple programs, serve multiple users on multiple computers geographically distributed around the world.

The simple OS is obviously inefficient as the process may have to wait for a data from a disk while the computer sits wasted. A very complex OS may have a significant overhead in figuring out what to do, so some machines sit wasted. A happy compromise has to be found to please the users and utilize the available resources.

The OS may support various kinds of processing such as

- batch processing (jobs or similar types are grouped)
- transactions processing (operations such as e-commerce purchases or bank transfers are supported)
- interactive processing (e.g., the entire web computing or games belong to this category).

There are many analogies to operating systems from daily life which help to understand its function in a computer system:

OS is a government - it facilitates management of resources, but does not produce actual work which is performed by application programs.

OS is a factory manager - as in a factory the manager facilitates the flow of work but does not produce any goods.

These two views are depicted in Fig. 5 where applications programs perform so called "useful work" while the OS so called "semi-useful work".

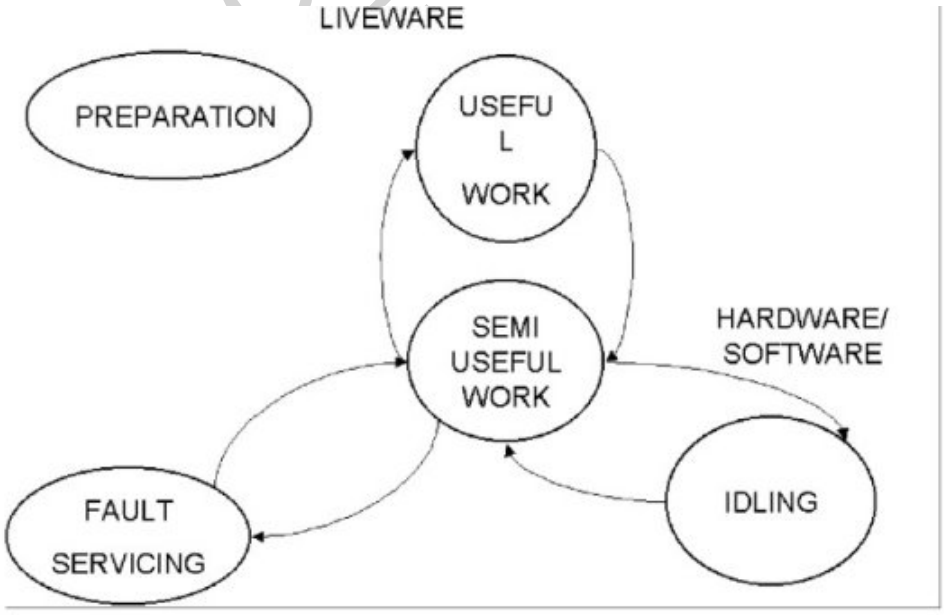

Figure 5: A workflow view of computer system.

OS is a magician - it hides software and hardware resources and gives the user appearance of problem-free world without typical hardware limitations (more of a daydreaming than the reality).

OS as stop lights - as the stop lights regulate the flow of cars the OS regulates the flow of processes (input/output operations, interrupts, memory protection etc.).
OS as a stock-exchange speculator - learns from the past mistakes to maximize profits (performance) in the future.

OS as a switching exchange - connects users, programs and hardware using processes (transactions).

OS as a middleman – serves as a broker between hardware and the operating system.

Summarizing this section it can be observed that the OS provides a virtual machine on top of the computer hardware that is easier to use than hardware alone. The OS accomplishes this task by: management of physical and virtual resources providing mechanisms and policies for the control of resources (both hardware and software), controlling of access, priorities and interactions and assuring quality-of-service (security, dependability, etc.).

To hardware resources to be managed belong control processing unit(s), main memory, secondary storage (disks, tapes, CD's), networks, input/output devices (keyboards, mouse, joystick, cameras, printers, speakers, etc.).

-
-
-

Bibliography

Peterson J. and Silberschatz A. (1983) *Operating Systems Concepts*, Reading, Addison-Wesley Publishing Company, Inc.

Silberschatz A. and Gavin P.B. (1994) *Operating Systems Concepts*, Reading, Addison-Wesley Publishing Company, Inc.

Solomon D.A., (1998) *Inside Windows NT*, 2nd Edition, MS Press

Tanenbaum A. S. (1995) *Distributed Operating Systems*, Englewood Cliffs, Prentice-Hall.

Tanenbaum A.S. (1992) *Modern Operating Systems*, Englewood Cliffs, Prentice-Hall.