

# INTRODUCTION TO OPERATING SYSTEMS\*

Duong Anh Duc

This work is produced by OpenStax-CNX and licensed under the  
Creative Commons Attribution License 3.0<sup>†</sup>

## Abstract

### Introduction to Operating Systems

Operating system is a hard term to define. Silberschatz et. al. defines that “**an operating system is a program that manages the computer hardware.**” However, what you consider an operating system depends on your needs and your view of the system. The first view is that operating system is a scheduler/simulator, on this view, the operating system has resources for which it is in charge, responsible for handing them out as well as recovering them later. Resources here include CPU, memory, I/O devices, and disk space. The second view is that operating system is a virtual machine, it means operating system provides a new machine. This machine could be the same as the underlying machine. Allows many users to believe they have an entire piece of hardware to themselves. This could implement a different, perhaps more powerful machine. Or just a different machine entirely. It may be useful to be able to completely simulate another machine with your current hardware. The other view is that operating system is a multiplexor which allows sharing of resources, provides protection from interference, and provides for a level of cooperation between users. This also for the economic reasons, we can't afford for all resources.

According to these three views, if we have enough hardware to give anyone too much; the hardware was well defined; the sharing problem is solved then we would not need operating systems. Unfortunately, they will still be needed as a servant or provider of services: need to provide things like in the above views, but deal with environments that are less than perfect, need to help the users use the computer by: providing commonly used subroutines; providing access to hardware facilities; providing higher-level "abstract" facilities; providing an environment which is easy, pleasant, and productive to use. This view as a provider of services fits well with our modern network view of computing, where most resources are services.

Internally, operating systems vary greatly in their makeup, since they are organized along many different lines. The design of a new operating system is a major task. It is important that the goals of the system be well defined before the design begins. These goals form the basis for choices among various algorithms and strategies.

Because an operating system is large and complex, it must be created piece by piece. Each of these pieces should be well delineated portion of the system, with carefully defined inputs, outputs, and functions.

## 1 What are the desires of an operating system?

We can discuss them in term of: Usability, Facilities, Cost, and Adaptability.

---

\*Version 1.1: Jul 7, 2009 1:33 am -0500

<sup>†</sup><http://creativecommons.org/licenses/by/3.0/>

### 1.1 Usability:

- Robustness

OS accept all valid inputs without error, and gracefully handle all invalid inputs. OS would not be crashed in any circumstances, and could be recovered in the case that we suddenly remove hardware while they are running, or lost of power supplied.

- Consistency

For example, if "-" means options flags in one place, it means it in another. The key idea is **conventions**. We should base on the concept: The Principle of Least Astonishment. This helps us easy to understand and adapt with the new system.

- Proportionality Simple, cheap and frequent things are easy. Also, expensive and disastrous things are hard.
- Forgiving Errors can be recovered from. Reasonable error messages. Example from "rm"; UNIX vs. TOPS.
- Convenient Not necessary to repeat things, or do awkward procedures to accomplish things. Example copying a file took a batch job.
- Powerful Has high level facilities.

### 1.2 Facilities

- The system should supply sufficient for intended use
- The facilities is complete, don't leave out any part of a facility.
- Appropriate, e.g. do not use fixed-width field input from terminal

### 1.3 Cost

- Want low cost and efficient services.
- Good algorithms. Make use of space/time tradeoffs, special hardware.
- Low overhead. Cost of doing nothing should be low. E.g., idle time at a terminal.
- Low maintenance cost. System should not require constant attention.

### 1.4 Adaptability

- Tailored to the environment. Support necessary activities. Do not impose unnecessary restrictions. What are the things people do most – make them easy.
- Changeable over time. Adapt as needs and resources change. E.g., expanding memory and new devices, or new user population.
- Extendible-Extensible: Adding new facilities and features - which look like the old ones.

## 2 Two main perspectives of an operating system

- Outside view: whether your system can support for those kinds of program, do they have many facilities: compiler, database, do they easy to use! At this level we focus on what services the system provides.
- Inside view: related to the internals, code, data structures. This is the system programmer view of an operating system. At this level you understand not only what is provided, but how it is provided.

## 3 Five main components of an operating system

### 3.1 Process management

**Process is a system abstraction, it illustrates that system has only one job to do.** Every program running on a computer, be it background services or applications, is a process. As long as a von Neumann architecture is used to build computers, only one process per CPU can be run at a time. Older microcomputer OSes such as MS-DOS did not attempt to bypass this limit, with the exception of interrupt processing, and only one process could be run under them. Mainframe operating systems have had multitasking capabilities since the early 1960s. Modern operating systems enable concurrent execution of many processes at once via multitasking even with one CPU. Process management is an operating system's way of dealing with running multiple processes. Since most computers contain one processor with one core, multitasking is done by simply switching processes quickly. Depending on the operating system, as more processes run, either each time slice will become smaller or there will be a longer delay before each process is given a chance to run. Process management involves computing and distributing CPU time as well as other resources. Most operating systems allow a process to be assigned a priority which affects its allocation of CPU time. Interactive operating systems also employ some level of feedback in which the task with which the user is working receives higher priority. Interrupt driven processes will normally run at a very high priority. In many systems there is a background process, such as the System Idle Process in Windows, which will run when no other process is waiting for the CPU.

### 3.2 Memory management

Current computer architectures arrange the computer's memory in a hierarchical manner, starting from the fastest registers, CPU cache, random access memory and disk storage. An operating system's memory manager coordinates the use of these various types of memory by tracking which one is available, which is to be allocated or deallocated and how to move data between them. This activity, usually referred to as virtual memory management, increases the amount of memory available for each process by making the disk storage seem like main memory. There is a speed penalty associated with using disks or other slower storage as memory – if running processes require significantly more RAM than is available, the system may start thrashing. This can happen either because one process requires a large amount of RAM or because two or more processes compete for a larger amount of memory than is available. This then leads to constant transfer of each process's data to slower storage.

Another important part of memory management is managing virtual addresses. If multiple processes are in memory at once, they must be prevented from interfering with each other's memory (unless there is an explicit request to utilise shared memory). This is achieved by having separate address spaces. Each process sees the whole virtual address space, typically from address 0 up to the maximum size of virtual memory, as uniquely assigned to it. The operating system maintains a page table that matches virtual addresses to physical addresses. These memory allocations are tracked so that when a process terminates, all memory used by that process can be made available for other processes.

The operating system can also write inactive memory pages to secondary storage. This process is called "paging" or "swapping" – the terminology varies between operating systems.

It is also typical for operating systems to employ otherwise unused physical memory as a page cache; requests for data from a slower device can be retained in memory to improve performance. The operating system can also pre-load the in-memory cache with data that may be requested by the user in the near future; SuperFetch is an example of this.

### 3.3 Disk and file systems

All operating systems include support for a variety of file systems.

Modern file systems comprise a hierarchy of directories. While the idea is conceptually similar across all general-purpose file systems, some differences in implementation exist. Two noticeable examples of this are

the character used to separate directories, and case sensitivity.

Unix demarcates its path components with a slash (/), a convention followed by operating systems that emulated it or at least its concept of hierarchical directories, such as Linux, Amiga OS and Mac OS X. MS-DOS also emulated this feature, but had already also adopted the CP/M convention of using slashes for additional options to commands, so instead used the backslash (\) as its component separator. Microsoft Windows continues with this convention; Japanese editions of Windows use ¥, and Korean editions use ₩. Versions of Mac OS prior to OS X use a colon (:) for a path separator. RISC OS uses a period (.).

Unix and Unix-like operating allow for any character in file names other than the slash, and names are case sensitive. Microsoft Windows file names are not case sensitive.

File systems may provide journaling, which provides safe recovery in the event of a system crash. A journaled file system writes information twice: first to the journal, which is a log of file system operations, then to its proper place in the ordinary file system. In the event of a crash, the system can recover to a consistent state by replaying a portion of the journal. In contrast, non-journaled file systems typically need to be examined in their entirety by a utility such as fsck or chkdsk. Soft update is an alternative to journaling that avoids the redundant writes by carefully ordering the update operations. Log-structured file systems and ZFS also differ from traditional journaled file systems in that they avoid inconsistencies by always writing new copies of the data, eschewing in-place updates.

Many Linux distributions support some or all of ext2, ext3, ReiserFS, Reiser4, GFS, GFS2, OCFS, OCFS2, and NILFS. Linux also has full support for XFS and JFS, along with the FAT file systems, and NTFS.

Microsoft Windows includes support for FAT12, FAT16, FAT32, and NTFS. The NTFS file system is the most efficient and reliable of the four Windows file systems, and as of Windows Vista, is the only file system which the operating system can be installed on. Windows Embedded CE 6.0 introduced ExFAT, a file system suitable for flash drives.

Mac OS X supports HFS+ as its primary file system, and it supports several other file systems as well, including FAT16, FAT32, NTFS and ZFS.

Common to all these (and other) operating systems is support for file systems typically found on removable media. FAT12 is the file system most commonly found on floppy discs. ISO 9660 and Universal Disk Format are two common formats that target Compact Discs and DVDs, respectively. Mount Rainier is a newer extension to UDF supported by Linux 2.6 kernels and Windows-Vista that facilitates rewriting to DVDs in the same fashion as what has been possible with floppy disks.

### 3.4 Networking

Most current operating systems are capable of using the TCP/IP networking protocols. This means that one system can appear on a network of the other and share resources such as files, printers, and scanners using either wired or wireless connections.

Many operating systems also support one or more vendor-specific legacy networking protocols as well, for example, SNA on IBM systems, DECnet on systems from Digital Equipment Corporation, and Microsoft-specific protocols on Windows. Specific protocols for specific tasks may also be supported such as NFS for file access.

### 3.5 Security

Many operating systems include some level of security. Security is based on the two ideas that:

- The operating system provides access to a number of resources, directly or indirectly, such as files on a local disk, privileged system calls, personal information about users, and the services offered by the programs running on the system;
- The operating system is capable of distinguishing between some requesters of these resources who are authorized (allowed) to access the resource, and others who are not authorized (forbidden). While

some systems may simply distinguish between "privileged" and "non-privileged", systems commonly have a form of requester identity, such as a user name. Requesters, in turn, divide into two categories:

- Internal security: an already running program. On some systems, a program once it is running has no limitations, but commonly the program has an identity which it keeps and is used to check all of its requests for resources.
- External security: a new request from outside the computer, such as a login at a connected console or some kind of network connection. To establish identity there may be a process of authentication. Often a username must be quoted, and each username may have a password. Other methods of authentication, such as magnetic cards or biometric data, might be used instead. In some cases, especially connections from the network, resources may be accessed with no authentication at all.

In addition to the allow/disallow model of security, a system with a high level of security will also offer auditing options. These would allow tracking of requests for access to resources (such as, "who has been reading this file?").

Security of operating systems has long been a concern because of highly sensitive data held on computers, both of a commercial and military nature. The United States<sup>1</sup> Government<sup>2</sup> Department of Defense (DoD) created the Trusted Computer System Evaluation Criteria (TCSEC) which is a standard that sets basic requirements for assessing the effectiveness of security. This became of vital importance to operating system makers, because the TCSEC was used to evaluate, classify and select computer systems being considered for the processing, storage and retrieval of sensitive or classified information.

---

<sup>1</sup>[http://en.wikipedia.org/wiki/United\\_States](http://en.wikipedia.org/wiki/United_States)

<sup>2</sup>[http://en.wikipedia.org/wiki/Government\\_of\\_the\\_United\\_States](http://en.wikipedia.org/wiki/Government_of_the_United_States)

## 4 Operating systems Market share 2007

---

<b>Client OS Market Share for June, 2007<sup>[1]</sup></b>
Windows XP - 81.94%
Windows Vista - 4.52%
Windows 2000 - 4.00%
Mac OS - 3.52%
MacIntel - 2.48%
Windows 98 - 1.14%
Linux - 0.71%
Windows NT - 0.66%
Windows Me - 0.59%
Nintendo Wii- 0.17%
Other - 0.20%

Figure 1

---

## 5 Questions to ask about operating systems.

### 5.1 Why are operating systems important?

- They consume more resources than any other program.

They may only use up a small percentage of the CPU time, but consider how many machines use the same program, all the time.

- They are the most complex programs.

They perform more functions for more users than any other program.

- They are necessary for any use of the computer.

When "the (operating) system" is down, the computer is down. Reliability and recovery from errors becomes critical.

- They are used by many users.

More hours of user time is spent dealing with the operating system. Visible changes in the operating system cause many changes to the users.

[1]<sup>3 4</sup> <http://marketshare.hitslink.com/report.aspx?qprid=2&qpmr=15&qpdt=1&qpct=3&qptimeframe=M&qpsp=101><sup>5</sup>

## 5.2 Why are operating systems difficult to create, use, and maintain?

- Size - too big for one person

Current systems have many millions lines of code. Involve 10-100 person years to build.

- Lifetime - the systems remain around longer than the programmers who wrote them.

The code is written and rewritten. Original intent is forgotten (UNIX was designed to be cute, little system - now 2 volumes this thick). Bug curve should be decreasing; but actually periodic - draw.

- Complexity - the system must do difficult things.

Deal with ugly I/O devices, multiplexing-juggling act, handle errors (**hard!**).

- Asynchronous - must do several things at once.

Handles interrupts, and must change what it is doing thousands of times a second - and still get work done.

- General purpose - must do many different things.

Run Doom, Java, Fortran, Lisp, Trek, Databases, Web Servers, etc. Everybody wants their stuff to run well.

## 6 History of Operating Systems

1. Single user (no OS).
2. Batch, uniprogrammed, run to completion.
  - The OS now must be protected from the user program so that it is capable of starting (and assisting) the next program in the batch.
3. Multiprogrammed
  - The purpose was to overlap CPU and I/O
  - Multiple batches
    - IBM OS/MFT (Multiprogramming with a Fixed number of Tasks)
      - \* OS for IBM system 360.
      - \* The (real) memory is partitioned and a batch is assigned to a fixed partition.
      - \* The memory assigned to a partition does not change.
      - \* Jobs were **spooled** from cards into the memory by a separate processor (an IBM 1401). Similarly output was spooled from the memory to a printer (a 1403) by the 1401.
    - IBM OS/MVT (Multiprogramming with a Variable number of Tasks) (then other names)

<sup>3</sup><http://cnx.org/content/m27320/latest/file:///tmp/%5B1%5D%20http://marketshare.hitslink.com/report.aspx%3Fqprid=2&qpmr=15&qpdt>

<sup>4</sup><http://cnx.org/content/m27320/latest/file:///tmp/%5B1%5D%20http://marketshare.hitslink.com/report.aspx%3Fqprid=2&qpmr=15&qpdt>

<sup>5</sup><http://cnx.org/content/m27320/latest/file:///tmp/%5B1%5D%20http://marketshare.hitslink.com/report.aspx%3Fqprid=2&qpmr=15&qpdt>

- \* Each job gets just the amount of memory it needs. That is, the partitioning of memory changes as jobs enter and leave
- \* MVT is a more “efficient” user of resources, but is more difficult.
- \* When we study memory management, we will see that, with varying size partitions, questions like compaction and “holes” arise.
- Time sharing
  - This is multiprogramming with rapid switching between jobs (processes). Deciding when to switch and which process to switch to is called scheduling.
  - We will study scheduling when we do processor management

#### 4. Personal Computers

- Serious PC Operating systems such as linux, Windows NT/2000/XP and (the newest) MacOS are multiprogrammed OSES.
- GUIs have become important. Debate as to whether it should be part of the kernel.
- Early PC operating systems were uniprogrammed and their direct descendants in some sense still are (e.g. Windows ME).

## 7 Operating systems are an unsolved problem.

Most of OS do not work very well, it crash too often, too slow, awkward to use, etc. Usually they do not do everything they were designed to do. They do not adapt to changes very well, e.g new devices, processors, applications. There are no perfect models to emulate.