

Introduction



Chapter 1 introduces the general topic of operating systems and a handful of important concepts (multiprogramming, time sharing, distributed system, and so on). The purpose is to show *why* operating systems are what they are by showing *how* they developed. In operating systems, as in much of computer science, we are led to the present by the paths we took in the past, and we can better understand both the present and the future by understanding the past.

Additional work that might be considered is learning about the particular systems that the students will have access to at your institution. This is still just a general overview, as specific interfaces are considered in Chapter 3.

- 1.1 A network computer relies on a centralized computer for most of its services. It can therefore have a minimal operating system to manage its resources. A personal computer on the other hand has to be capable of providing all of the required functionality in a stand-alone manner without relying on a centralized manner. Scenarios where administrative costs are high and where sharing leads to more efficient use of resources are precisely those settings where network computers are preferred.
- 1.2
 - a. **A dormitory floor**—A LAN.
 - b. **A university campus**—A LAN, possibly a WAN for very large campuses.
 - c. **A state**—A WAN.
 - d. **A nation**—A WAN.
- 1.3 Caches are useful when two or more components need to exchange data, and the components perform transfers at differing speeds. Caches solve the transfer problem by providing a buffer of intermediate speed between the components. If the fast device finds the data it needs in the cache, it need not wait for the slower device. The data in the cache must be kept consistent with the data in the components. If a component has a data value change, and the datum is also in the cache, the cache must also be updated. This is especially a problem on multiprocessor systems where more than one process may be accessing a datum. A component

may be eliminated by an equal-sized cache, but only if: (a) the cache and the component have equivalent state-saving capacity (that is, if the component retains its data when electricity is removed, the cache must retain data as well), and (b) the cache is affordable, because faster storage tends to be more expensive.

- 1.4 When there are few other users, the task is large, and the hardware is fast, time-sharing makes sense. The full power of the system can be brought to bear on the user's problem. The problem can be solved faster than on a personal computer. Another case occurs when lots of other users need resources at the same time.

A personal computer is best when the job is small enough to be executed reasonably on it and when performance is sufficient to execute the program to the user's satisfaction.

- 1.5 The four steps are:
 - a. Reserve machine time.
 - b. Manually load program into memory.
 - c. Load starting address and begin execution.
 - d. Monitor and control execution of program from console.
- 1.6 The distinction between kernel mode and user mode provides a rudimentary form of protection in the following manner. Certain instructions could be executed only when the CPU is in kernel mode. Similarly, hardware devices could be accessed only when the program is executing in kernel mode. Control over when interrupts could be enabled or disabled is also possible only when the CPU is in kernel mode. Consequently, the CPU has very limited capability when executing in user mode, thereby enforcing protection of critical resources.
- 1.7
 - a. Stealing or copying one's programs or data; using system resources (CPU, memory, disk space, peripherals) without proper accounting.
 - b. Probably not, since any protection scheme devised by humans can inevitably be broken by a human, and the more complex the scheme, the more difficult it is to feel confident of its correct implementation.
- 1.8 The processor could keep track of what locations are associated with each process and limit access to locations that are outside of a program's extent. Information regarding the extent of a program's memory could be maintained by using base and limits registers and by performing a check for every memory access.
- 1.9 Handheld computers are much smaller than traditional desktop PCs. This results in smaller memory, smaller screens, and slower processing capabilities than a standard desktop PC. Because of these limitations, most handhelds currently can perform only basic tasks such as calendars, email, and simple word processing. However, due to their small size, they are quite portable and, when they are equipped with

wireless access, can provide remote access to electronic mail and the world wide web.

- 1.10** The client-server model firmly distinguishes the roles of the client and server. Under this model, the client requests services that are provided by the server. The peer-to-peer model doesn't have such strict roles. In fact, all nodes in the system are considered peers and thus may act as *either* clients or servers—or both. A node may request a service from another peer, or the node may in fact provide such a service to other peers in the system.

For example, let's consider a system of nodes that share cooking recipes. Under the client-server model, all recipes are stored with the server. If a client wishes to access a recipe, it must request the recipe from the specified server. Using the peer-to-peer model, a peer node could ask other peer nodes for the specified recipe. The node (or perhaps nodes) with the requested recipe could provide it to the requesting node. Notice how each peer may act as both a client (it may request recipes) and as a server (it may provide recipes).

- 1.11** An operating system for a machine of this type would need to remain in control (or monitor mode) at all times. This could be accomplished by two methods:
- a. Software interpretation of all user programs (like some BASIC, Java, and LISP systems, for example). The software interpreter would provide, in software, what the hardware does not provide.
 - b. Require that all programs be written in high-level languages so that all object code is compiler-produced. The compiler would generate (either in-line or by function calls) the protection checks that the hardware is missing.
- 1.12** Generally, operating systems for batch systems have simpler requirements than for personal computers. Batch systems do not have to be concerned with interacting with a user as much as a personal computer. As a result, an operating system for a PC must be concerned with response time for an interactive user. Batch systems do not have such requirements. A pure batch system also may have not to handle time sharing, whereas an operating system must switch rapidly between different jobs.
- 1.13** The following operations need to be privileged: Set value of timer, clear memory, turn off interrupts, modify entries in device-status table, access I/O device. The rest can be performed in user mode.
- 1.14** In single-processor systems, the memory needs to be updated when a processor issues updates to cached values. These updates can be performed immediately or in a lazy manner. In a multiprocessor system, different processors might be caching the same memory location in its local caches. When updates are made, the other cached locations need to be invalidated or updated. In distributed systems, consistency of cached memory values is not an issue. However, consistency problems might arise when a client caches file data.

- 1.15 Open source operating systems have the advantages of having many people working on them, many people debugging them, ease of access and distribution, and rapid update cycles. Further, for students and programmers there is certainly an advantage to being able to view and modify the source code. Typically open source operating systems are free for some forms of use, usually just requiring payment for support services. Commercial operating system companies usually do not like the competition that open source operating systems bring because these features are difficult to compete against. Some open source operating systems do not offer paid support programs. Some companies avoid open source projects because they need paid support, so that they have some entity to hold accountable if there is a problem or they need help fixing an issue. Finally, some complain that a lack of discipline in the coding of open source operating systems means that backward-compatibility is lacking making upgrades difficult, and that the frequent release cycle exacerbates these issues by forcing users to upgrade frequently.
- 1.16 Clustered systems are typically constructed by combining multiple computers into a single system to perform a computational task distributed across the cluster. Multiprocessor systems on the other hand could be a single physical entity comprising of multiple CPUs. A clustered system is less tightly coupled than a multiprocessor system. Clustered systems communicate using messages, while processors in a multiprocessor system could communicate using shared memory.
- In order for two machines to provide a highly available service, the state on the two machines should be replicated and should be consistently updated. When one of the machines fails, the other could then takeover the functionality of the failed machine.
- 1.17 The main difficulty is keeping the operating system within the fixed time constraints of a real-time system. If the system does not complete a task in a certain time frame, it may cause a breakdown of the entire system it is running. Therefore when writing an operating system for a real-time system, the writer must be sure that his scheduling schemes don't allow response time to exceed the time constraint.
- 1.18 The CPU can initiate a DMA operation by writing values into special registers that can be independently accessed by the device. The device initiates the corresponding operation once it receives a command from the CPU. When the device is finished with its operation, it interrupts the CPU to indicate the completion of the operation.
- Both the device and the CPU can be accessing memory simultaneously. The memory controller provides access to the memory bus in a fair manner to these two entities. A CPU might therefore be unable to issue memory operations at peak speeds since it has to compete with the device in order to obtain access to the memory bus.
- 1.19 For real-time systems, the operating system needs to support virtual memory and time sharing in a fair manner. For handheld systems, the operating system needs to provide virtual memory, but does not need to provide time-sharing. Batch programming is not necessary in both settings.

- 1.20 Although most systems only distinguish between user and kernel modes, some CPUs have supported multiple modes. Multiple modes could be used to provide a finer-grained security policy. For example, rather than distinguishing between just user and kernel mode, you could distinguish between different types of user mode. Perhaps users belonging to the same group could execute each other's code. The machine would go into a specified mode when one of these users was running code. When the machine was in this mode, a member of the group could run code belonging to anyone else in the group.

Another possibility would be to provide different distinctions within kernel code. For example, a specific mode could allow USB device drivers to run. This would mean that USB devices could be serviced without having to switch to kernel mode, thereby essentially allowing USB device drivers to run in a quasi-user/kernel mode.

- 1.21
- a. **Batch.** Jobs with similar needs are batched together and run through the computer as a group by an operator or automatic job sequencer. Performance is increased by attempting to keep CPU and I/O devices busy at all times through buffering, off-line operation, spooling, and multiprogramming. Batch is good for executing large jobs that need little interaction; they can be submitted and picked up later.
 - b. **Interactive.** This system is composed of many short transactions where the results of the next transaction may be unpredictable. Response time needs to be short (seconds) since the user submits and waits for the result.
 - c. **Time sharing.** This system uses CPU scheduling and multiprogramming to provide economical interactive use of a system. The CPU switches rapidly from one user to another. Instead of having a job defined by spooled card images, each program reads its next control card from the terminal, and output is normally printed immediately to the screen.
 - d. **Real time.** Often used in a dedicated application, this system reads information from sensors and must respond within a fixed amount of time to ensure correct performance.
 - e. **Network.** Provides operating system features across a network such as file sharing.
 - f. **SMP.** Used in systems where there are multiple CPUs each running the same copy of the operating system. Communication takes place across the system bus.
 - g. **Distributed.** This system distributes computation among several physical processors. The processors do not share memory or a clock. Instead, each processor has its own local memory. They communicate with each other through various communication lines, such as a high-speed bus or local area network.
 - h. **Clustered.** A clustered system combines multiple computers into a single system to perform computational tasks distributed across the cluster.

- i. **Handheld.** A small computer system that performs simple tasks such as calendars, email, and web browsing. Handheld systems differ from traditional desktop systems with smaller memory and display screens and slower processors.
- 1.22 Symmetric multiprocessing treats all processors as equals, and I/O can be processed on any CPU. Asymmetric multiprocessing has one master CPU and the remainder CPUs are slaves. The master distributes tasks among the slaves, and I/O is usually done by the master only. Multiprocessors can save money by not duplicating power supplies, housings, and peripherals. They can execute programs more quickly and can have increased reliability. They are also more complex in both hardware and software than uniprocessor systems.
- 1.23
 - a. Mainframes: memory and CPU resources, storage, network bandwidth
 - b. Workstations: memory and CPU resources
 - c. Handheld computers: power consumption, memory resources
- 1.24 An interrupt is a hardware-generated change of flow within the system. An interrupt handler is summoned to deal with the cause of the interrupt; control is then returned to the interrupted context and instruction. A trap is a software-generated interrupt. An interrupt can be used to signal the completion of an I/O to obviate the need for device polling. A trap can be used to call operating system routines or to catch arithmetic errors.
- 1.25 Say processor 1 reads data *A* with value 5 from main memory into its local cache. Similarly, processor 2 reads data *A* into its local cache as well. Processor 1 then updates *A* to 10. However, since *A* resides in processor 1's local cache, the update only occurs there and not in the local cache for processor 2.
- 1.26 Consider the following two alternatives: **asymmetric clustering** and **parallel clustering**. With asymmetric clustering, one host runs the database application with the other host simply monitoring it. If the server fails, the monitoring host becomes the active server. This is appropriate for providing redundancy. However, it does not utilize the potential processing power of both hosts. With parallel clustering, the database application can run in parallel on both hosts. The difficulty in implementing parallel clusters is providing some form of distributed locking mechanism for files on the shared disk.