

Chapter Summaries from the book

Operating System Concepts / Chapter 1

An operating system is software that manages the computer hardware, as well as providing an environment for application programs to run. Perhaps the most visible aspect of an operating system is the interface to the computer system it provides to the human user.

For a computer to do its job of executing programs, the programs must be in main memory. Main memory is the only large storage area that the processor can access directly. It is an array of bytes, ranging in size from millions to billions. Each byte in memory has its own address.

The main memory is usually a volatile storage device that loses its contents when power is turned off or lost. Most computer systems provide secondary storage as an extension of main memory. Secondary storage provides a form of nonvolatile storage that is capable of holding large quantities of data permanently.

The wide variety of storage systems in a computer system can be organized in a hierarchy according to speed and cost. The higher levels are expensive, but they are fast. As we move down the hierarchy, the cost per bit generally decreases, whereas the access time generally increases.

There are several different strategies for designing a computer system. Single-processor systems have only one processor, while multiprocessor systems contain two or more processors that share physical memory and peripheral devices.

The most common multiprocessor design is symmetric multiprocessing (or SMP), where all processors are considered peers and run independently of one another. Clustered systems are a specialized form of multiprocessor systems and consist of multiple computer systems connected by a local-area network.

To best utilize the CPU, modern operating systems employ multiprogramming, which allows several jobs to be in memory at the same time, thus ensuring that the CPU always has a job to execute. Time-sharing systems are an extension of multiprogramming wherein CPU scheduling algorithms rapidly switch between jobs, thus providing the illusion that each job is running concurrently.

The operating system must ensure correct operation of the computer system. To prevent user programs from interfering with the proper operation of the system, the hardware has two modes:

user mode and kernel mode. Various instructions (such as I/O instructions and halt instructions) are privileged and can be executed only in kernel mode.

The memory in which the operating system resides must also be protected from modification by the user. A timer prevents infinite loops. These facilities (dual mode, privileged instructions, memory protection, and timer interrupt) are basic building blocks used by operating systems to achieve correct operation.

A process (or job) is the fundamental unit of work in an operating system. Process management includes creating and deleting processes and providing mechanisms for processes to communicate and synchronize with each other.

An operating system manages memory by keeping track of what parts of memory are being used and by whom. The operating system is also responsible for dynamically allocating and freeing memory space. Storage space is also managed by the operating system; this includes providing file systems for representing files and directories and managing space on mass-storage devices.

Operating systems must also be concerned with protecting and securing the operating system and users. Protection measures control the access of processes or users to the resources made available by the computer system. Security measures are responsible for defending a computer system from external or internal attacks.

Several data structures that are fundamental to computer science are widely used in operating systems, including lists, stacks, queues, trees, hash functions, maps, and bitmaps.

Operating System Structures / Chapter 2

Operating systems provide a number of services. At the lowest level, system calls allow a running program to make requests from the operating system directly. At a higher level, the command interpreter or shell provides a mechanism for a user to issue a request without writing a program. Commands may come from files during batch-mode execution or directly from a terminal or desktop GUI when in an interactive or time-shared mode. System programs are provided to satisfy many common user requests.

The types of requests vary according to level. The system-call level must provide the basic functions, such as process control and file and device manipulation. Higher-level requests, satisfied by the command interpreter or system programs, are translated into a sequence of system calls. System services can be classified into several categories: program control, status requests, and I/O requests. Program errors can be considered implicit requests for service.

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. The type of system desired is the foundation for choices among various algorithms and strategies that will be needed.

Throughout the entire design cycle, we must be careful to separate policy decisions from implementation details (mechanisms). This separation allows maximum flexibility if policy decisions are to be changed later.

Once an operating system is designed, it must be implemented. Operating systems today are almost always written in a systems-implementation language or in a higher-level language. This feature improves their implementation, maintenance, and portability.

A system as large and complex as a modern operating system must be engineered carefully. Modularity is important. Designing a system as a sequence of layers or using a microkernel is considered a good technique. Many operating systems now support dynamically loaded modules, which allow adding functionality to an operating system while it is executing.

Generally, operating systems adopt a hybrid approach that combines several different types of structures.

Debugging process and kernel failures can be accomplished through the use of debuggers and other tools that analyze core dumps. Tools such as DTrace analyze production systems to find bottlenecks and understand other system behavior.

To create an operating system for a particular machine configuration, we must perform system generation. For the computer system to begin running, the CPU must initialize and start executing the bootstrap program in firmware. The bootstrap can execute the operating system directly if the operating system is also in the firmware, or it can complete a sequence in which it loads progressively smarter programs from firmware and disk until the operating system itself is loaded into memory and executed.

Part Two

Process Management

A process can be thought of as a program in execution. A process will need certain resources — such as CPU time, memory, files, and I/O devices — to accomplish its task. These resources are allocated to the process either when it is created or while it is executing.

A process is the unit of work in most systems. Systems consist of a collection of processes: operating-system processes execute system code, and user processes execute user code. All these processes may execute concurrently.

Although traditionally a process contained only a single thread of control as it ran, most modern operating systems now support processes that have multiple threads.

The operating system is responsible for several important aspects of process and thread management: the creation and deletion of both user and system processes; the scheduling of processes; and the provision of mechanisms for synchronization, communication, and deadlock handling for processes.

Processes / Chapter 3

Each process is represented in the operating system by a process control block (PCB) - also called a task control block.

- Process state. The state may be new, ready, running, waiting, halted, and so on.
- Program counter. The counter indicates the address of the next instruction to be executed for this process.
- CPU registers. The registers vary in number and type, depending on the computer architecture. They include accumulators, index registers, stack pointers, and general-purpose registers, plus any condition-code information. Along with the program counter, this state information must be saved when an interrupt occurs, to allow the process to be continued correctly afterward (Figure 3.4).
- CPU - scheduling information. This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters. (Chapter 6 describes process scheduling.)
- Memory-management information. This information may include such items as the value of the base and limit registers and the page tables, or the segment tables, depending on the memory system used by the operating system