

Preface

The primary objective of a first course in Operating Systems is to develop an understanding of the fundamental concepts and techniques of operating systems. Today, a large section of students is already exposed to diverse information on operating systems due to practical exposure to operating systems and literature on the Internet; such students have a lot of information but few concepts about operating systems. This situation makes teaching of operating systems concepts a challenging task because it is necessary to retrofit some concepts to the information presented by these students without boring them, yet do it in a manner that introduces concepts to first time learners of operating systems without intimidating them. This book presents operating system concepts and techniques in a manner that incorporates these requirements.

General Approach

The book begins by building a core knowledge of what makes an operating system tick in Chapter 2. It presents an operating system as an intermediary between a computer system and user computations whose task is to provide good service to users and achieve efficient use of the computer system. A discussion of interaction of an operating system with the computer system on one hand and with user computations on the other hand consolidates this view and adds practical details to it. This approach has the effect of demystifying an operating system for a new reader, and relating to the background of an experienced reader. It also emphasizes key features of computer architecture which are essential for a study of operating systems. This part of the book provides a basis for discussing details of functions performed by an operating system in more detail.

The rest of the book follows an analogous approach. Each chapter identifies fundamental concepts involved in some functionality of an operating system, describes relevant features in computer architecture, discusses relevant operating system techniques and illustrates their operation through examples. The highlights of this approach are:

- Fundamental concepts are introduced in simple terms.
- Numerous examples are included to illustrate concepts and techniques.
- Implementation details and case studies are organized as small capsules spread throughout the text.
- Optional sections are devoted to advanced topics, e.g. deadlock characterization, kernel memory allocation, synchronization and scheduling in multiprocessor systems, file sharing semantics, file system reliability and capabilities.

The key advantage of this approach is that concepts, techniques and case studies are integrated into cohesive chapters. Hence many design and implementation details look 'obvious' when the reader encounters them. This fact helps to emphasize that the study of operating systems must be based on a sound understanding of concepts. This is the most important message a text can give to a student of operating systems who will face both a rich diversity and rapid changes in features of operating systems during his/her career.

Pedagogic Features

Part I

FUNDAMENTAL CONCEPTS

An operating system controls use of a computer system's resources such as CPUs, memory, and I/O devices to meet computational requirements of its users. Users expect convenience, quality of service, and security while executing their programs, whereas system administrators expect efficient use of the computer's resources and good performance in executing user programs. These diverse expectations can be characterized as *user convenience*, *security* and *efficient use of resources*; they form the primary goals of an operating system. The extent to which an operating system meets each of these goals depends on its *computing environment*, i.e., the computer system's hardware, its interfaces with other computers, and the nature of computations performed by its users.

Part Introduction

Each part of the book begins with a description of its contents, and a road map of the chapters in the part.

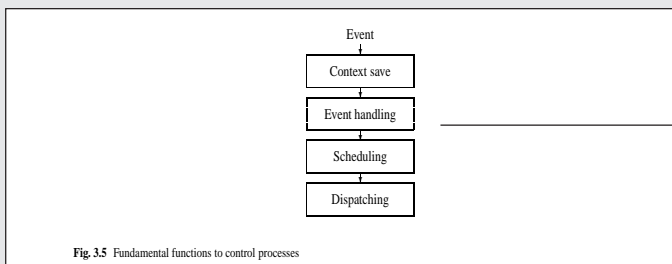


Fig. 3.5 Fundamental functions to control processes

Figures and Boxes

Figures depict practical arrangements used to handle user computations and use of resources, stepwise operation of specific techniques, or comparison of alternative techniques to project their strengths and weaknesses. Boxes are used to enclose key features of operating system functionalities being discussed. They also serve as overviews or summaries of specific topics.

Sequence of single programs Each single program in the sequence is initiated by the user through a separate command. However, a sequence of single programs has its own semantics—a program should be executed only if the previous programs in the sequence executed successfully. To achieve a common goal, the programs must explicitly interface their inputs and outputs with other programs in the sequence. Example 1.1 discusses the notion of a sequence of single programs in MS DOS.

Example 1.1 An MS DOS .bat file can be used to form a sequence of single programs. The file contains a sequence of commands, each command indicating execution of a program. Let a .bat file contain commands to initiate a sequence of programs to compile, link and execute a C program: The C compiler compiles a C program existing in file `alpha.c`, `linker` links the output of the compiler to generate an executable program named `demo`, and program `demo` executes on the data contained in file `sample`. The command interpreter loads the C compiler into the memory for execution. When the compiler completes its operation, the command interpreter initiates execution of the linker. Eventually, it initiates execution of `demo`.

Examples

Examples demonstrate the key issues concerning concepts and techniques being discussed. Examples are typeset in a different style to set them apart from the main body of the text. This feature helps a reader to skip an example if he/she does not want the flow of ideas to be interrupted, especially while reading a chapter for the first time.

Program Code

Program code is presented in an easy to understand pseudo-code form.

```

size : integer value (...);
buffer: array[1..size] of ...;
copy_sample, disk_write, housekeeping : string;
no_of_samples : integer;
/* Create processes */
copy_sample := create_process(move_to_buffer(), 3);
disk_write := create_process(write_to_disk(), 2);
housekeeping := create_process(analysis(), 1);
/* Send size information to copy_sample and disk_write */
send (copy_sample, size);
send (disk_write, size);
/* Check status of all processes */
over := false;
while (over = false)
  if status(copy_sample) = terminated and status(disk_write) =
    terminated and status(housekeeping) = terminated then
    over := true;
  terminate();
end;

procedure move_to_buffer();
buf_size : integer;
/* Signal handler for changing buf_size */

```

Exercises

Exercises are included at the end of each chapter. These include numerical problems based on material covered in the text, as well as challenging conceptual questions which test understanding of the concepts and techniques covered in a chapter and also provide deeper insights.

24 Operating Systems

EXERCISE 1

1. Give examples of two situations in which user convenience conflicts with efficient use of a computer system.
2. Resource preemption may be performed to provide (a) fairness in the use of resources, (b) more efficient use. Describe a situation in which resource preemption provides fairness in the use of resources. Read Chapter 2 and describe a situation in which preemption is used to obtain efficient use.
3. An OS designer makes the following policy statement: "Consider preemption of a resource from a program only if the program can resume its operation after the resource is re-granted to it as if the preemption had not occurred." Justify preemption of CPU and memory from a program. Also argue why preemption of a magnetic tape cartridge or printer is inadvisable.
4. In the Unix operating system, a print server implements printing of files as desired by users. When a user executes a command to print a file, the print server copies the file on the disk and enters it in a print queue. It is printed when its entry reaches the head of the print queue.

Chapter introduction The chapter introduction precedes the first section of each chapter. It describes the objective and importance of the chapter and describes the topics covered in the chapter.

Snapshots of concurrent systems Students have difficulty visualizing concurrent operation of processes in a software system. This difficulty leads to an inadequate understanding of process synchronization. A snapshot depicts the state of different processes and the synchronization data to provide a holistic view of activities in a concurrent system.

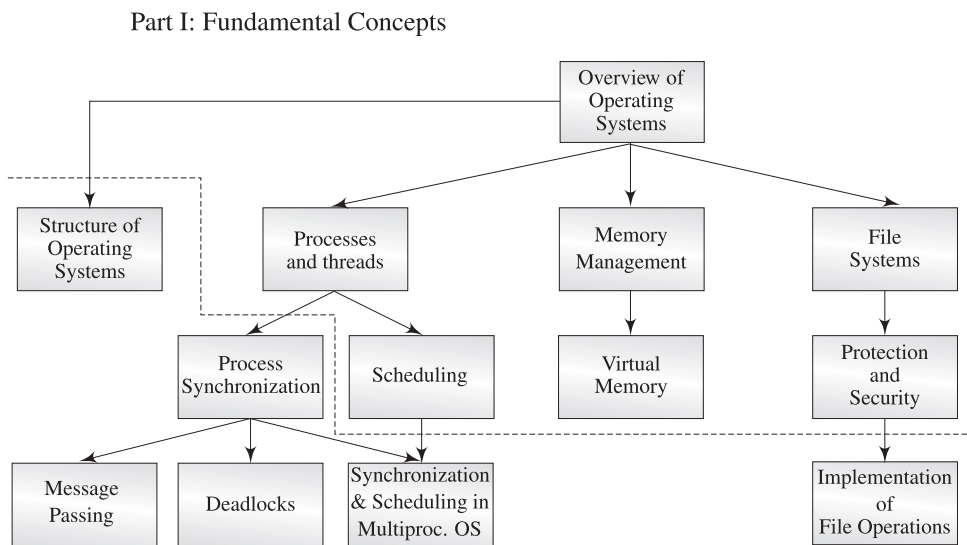
Optional sections Optional sections are devoted to advanced topics. These sections discuss theoretical basis for certain topics or advanced implementation issues. An instructor may include these sections in an offering of an operating system course depending on the availability of time, or selectively assign them for self-study.

Case studies Case studies are organized as independent sections. They illustrate practical issues, arrangements and tradeoffs in the design and implementation of an operating system. Different versions of the Unix and Windows operating systems, and Linux are the primary operating systems covered by the case studies.

Instructor resources A detailed solutions manual and slides for classroom usage are planned and will be available at <http://www.mhhe.com/dhamdhere/os>

Organization of the book

The book deals with both conventional operating systems for use in independent computer systems, and distributed operating systems for use in computer systems that are formed by networking independent computers. The introduction discusses the fundamental concerns of an operating system and describes different kinds of computations and different methods of ensuring efficient use of resources. The discussion of conventional operating systems is organized into two parts, devoted to fundamental concepts and advanced topics, respectively. The discussion of distributed operating systems forms the third part by itself. The structure of the parts and interdependency between chapters is as follows:



Part II: Advanced Topics

Part I: Fundamental Concepts The first part consists of seven chapters. Chapter 2 focuses on interaction of an operating system with a computer system and with user computations. It also describes different classes of operating systems and describes the fundamental concepts and techniques used by them.

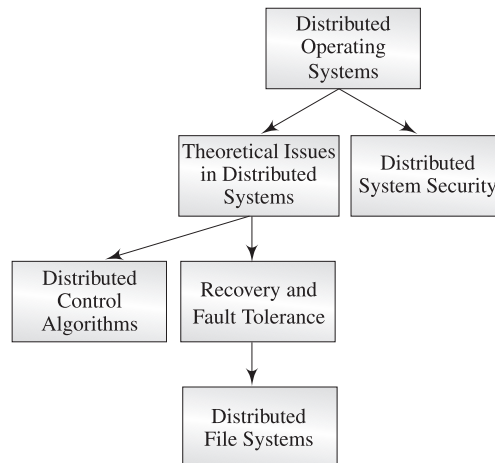
An operating system uses the concepts of *process* and *thread* to manage execution of programs—informally, both a process and a thread represent an execution of a program; we use *process* as a generic term. Chapter 3 describes the user and operating system views of processes, i.e., how processes are created, how they interact with one another, and how they are controlled by the operating system. An operating system overlaps execution of processes to provide good user service and ensure efficient use of resources. Chapter 4 describes the scheduling techniques used for this purpose.

The operating system shares the computer's memory between processes. Chapter 5 deals with the key issue of memory fragmentation, which is a situation in which an area of memory cannot be used because it is too small, and techniques which address memory fragmentation. It also discusses the techniques used by the kernel to manage its own data structures. Chapter 6 discusses implementation of virtual memory, which enables execution of a program whose size exceeds the size of memory.

The next two chapters deal with the File system, and security and protection of data and programs stored in files. Chapter 7 describes facilities for creation, access, sharing and reliable storage of files. Chapter 8 discusses how files are protected against illegal forms of access by users.

Part II: Advanced Concepts This part consists of six chapters devoted to advanced topics dealing with processes and their operation, implementation of file systems, and the structure of operating systems. Chapter 9 deals with process synchronization techniques that enable processes of an application to share common data or coordinate their activities to achieve a common goal, while Chapter 10 discusses how processes exchange messages to pass information to one another. Chapter 11 discusses deadlock, which is a situation in which processes wait for each other indefinitely, thereby, stalling an application. Chapter 12 describes how an OS manages I/O devices and describes techniques it uses to achieve efficient implementation of file operations. Chapter 13 deals with synchronization and scheduling in a multiprocessor operating system. Chapter 14 discusses operating system design techniques that facilitate easy modification of an operating system (1) for use on a computer system with a different architecture, or (2) to meet new requirements of its users.

Part III: Distributed Operating Systems A distributed operating system differs from a conventional one in that the resources, processes and control operations of the OS are distributed between different nodes. This difference gives rise to a host of issues concerning reliability, efficiency, consistency and security of computations and of the OS itself. This Part contains six small chapters that address these issues.



Part III: Distributed Operating Systems

Using this book

This book is intended as a text for a course on operating systems that is modeled after the IEEE and ACM curricula in Computer Science. The first part of the book covers the fundamental OS principles, so it is adequate for a light course on operating system principles by itself. The first two parts of the book and selected sections of Chapters 15–20 of the third part are together adequate for an operating system principles course in a Computer Science or related curriculum. However, all topics discussed here cannot be covered in a semester, so an instructor may wish to omit some of the sections on advanced topics or the chapters on message passing, synchronization and scheduling in multiprocessor operating systems, and structure of operating systems; or assign them for self-study. The three parts of the book together cover the topics needed for a two-course sequence on operating system principles. A course on Distributed operating systems, possibly at the postgraduate level, could use the third part of the book for a substantial portion of the course.

Apart from an introduction to computing, this book does not assume any specific background. Hence instructors and students are likely to find that it contains a lot of introductory material which students already know. I have included this material for one very important reason: As mentioned at the start of the preface, students know many *things* on their own, but often lack *concepts*. So it is useful for students to read even familiar material which is presented in a concept-based manner. For the same reason, I consider it essential for instructors to cover most of Chapter 2, and particularly the following topics, in class:

- Section 2.1: OS and the Computer System, particularly I/O, interrupt, and memory protection hardware; and system calls.
- Section 2.5: Multiprogramming, particularly program mix and priority.

Differences with the first edition

The book has been restructured substantially. Chapters in a part and their sequences are different from those in the first edition. Chapters and their sections have been reorganized, new topics have been added, and tables have been introduced to achieve better focus on concepts. A section containing a preview of the book has been added in the first chapter, and introductions have been added to parts and chapters to better motivate the reader. I hope students and instructors like the new format of the book.

D M DHAMDHARE