

# FOUNDATIONS OF COMPUTER TECHNOLOGY

A. John Anderson

CHAPMAN & HALL



***FOUNDATIONS OF  
COMPUTER TECHNOLOGY***

**A. John Anderson PhD C.Eng MIEE**

*Department of Electrical and Electronic Engineering  
University of Paisley, Paisley, UK*



**CHAPMAN & HALL**

London · Weinheim · New York · Tokyo · Melbourne · Madras

Published by Chapman & Hall, 2-6 Boundary Row, London SE1 8HN, UK

Chapman & Hall, 2-6 Boundary Row, London SE1 8HN, UK

Chapman & Hall GmbH, Pappelallee 3, 69469 Weinheim, Germany

Chapman & Hall USA, One Penn Plaza, 41st Floor, New York NY 10119, USA

Chapman & Hall Japan, ITP-Japan, Kyowa Building, 3F, 2-2-1 Hirakawacho  
Chiyoda-ku, Tokyo 102, Japan

Chapman & Hall Australia, Thomas Nelson Australia, 102 Dodds Street, South  
Melbourne, Victoria 3205, Australia

Chapman & Hall India, R. Seshadri, 32 Second Main Road, CIT East, Madras  
600 035, India

First edition 1994

Reprinted 1996, 1998

© 1994 A. John Anderson

Printed and bound in Croatia

ISBN 0 412 59810 8

Apart from any fair dealing for the purposes of research or private study, or criticism or review, as permitted under the UK Copyright Designs and Patents Act, 1988, this publication may not be reproduced, stored, or transmitted, in any form or by any means, without the prior permission in writing of the publishers, or in the case of reprographic reproduction only in accordance with the terms of the licences issued by the Copyright Licensing Agency in the UK, or in accordance with the terms of licences issued by the appropriate Reproduction Rights Organization outside the UK. Enquiries concerning reproduction outside the terms stated here should be sent to the publishers at the London address printed on this page.

The publisher makes no representation, express or implied, with regard to the accuracy of the information contained in this book and cannot accept any legal responsibility or liability for any errors or omissions that may be made.

A catalogue record for this book is available from the British Library

Library of Congress Catalog Card Number: 94-71829

The use of time sharing gives the illusion that a number of users are using the CPU simultaneously. As a result, each user sees what is called a virtual processor, giving the impression that he or she has sole access to the complete machine. In this way, by multiplexing the use of every real resource within the computer, the operating system turns individual real resources into a number of virtual resources. In order to provide this view of the computer, the operating system must hide the details of the management of the CPU and the other system resources from the user. To the user these virtual resources exhibit a behaviour much like that of the real resources but they do not need to worry about the complex control mechanisms that may be required to operate the resource, since these mechanisms are handled by the operating system.

Time sharing is probably the commonest form of operating system for large modern computers. In such systems there may be a large number of concurrent tasks, the majority of which are executing interactively in communication with user terminals, while other background batch programs may also be executed in the short idle intervals.

Time sharing systems typically have a very high volume direct access backing store to hold the large amount of user programs and files. They also require a very high speed data access and transfer mechanism since, when a terminal comes on line to request processing from the CPU, the program requested by the new user has to be read from the secondary memory into the main memory and after processing it has to be transferred back to the direct access store and the next program demanded called in. A sophisticated set of hardware and software mechanisms is therefore required to implement time sharing techniques. These mechanisms control the search for and the transfer of data to/from the terminals and also organize access to the wide range of programs and files held in secondary memory.

#### 10.1.4 REAL TIME OPERATING SYSTEM

A real time operating system (RTOS) is normally used to handle real time applications where jobs with higher priorities are executed before jobs with lower priorities. It can also be designed to handle batch and real time tasks concurrently, e.g. a typical mainframe computer might process a stream of batch programs fed into the computer from a disk or terminal while at the same time being able to respond promptly to a request from a number of user terminals.

The RTOS for this type of mainframe is basically a two-mode system where the foreground mode operates as an on-line program to control the data entry stations, while the background mode is batch orientated, compilation, file update, sorting and so on. Finally, an RTOS can be programmed to schedule jobs at some preset time, for example the preparation of an inventory report at the end of each working day.

### 10.2 DISK OPERATING SYSTEM (DOS)

The operating systems used in small micro and minicomputer systems are usually classified according to the nature of the storage devices that hold them. A tape operating system stores the main part of the program on tape, while a disk operating system (DOS) stores the bulk of its commands on a magnetic disk. All modern computers use disk operating systems. There are a number of popular disk operating systems including CP/M, MS-DOS, OS/2 and UNIX.

### 10.3 OPERATING SYSTEM FUNCTIONS

The basic goals for an operating system are to provide a programmer interface to the machine; provide mechanisms to implement programs; and to facilitate the matching of applications to the machine.

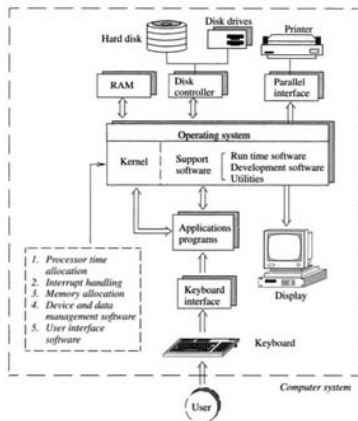
Figure 10.5 shows the relationship between the operating system, the system hardware, the application programs and the user. The various functions of a typical operating system are also noted.

The **kernel** of an operating system is the central part of the operating system. It is also known as the nucleus, monitor or executive, and is the part of the operating system which provides a series of basic utility routines, to observe, supervise, control and verify the operations of the computer system. Since the kernel is the most commonly used piece of software in any processing system it is typically held in primary storage, i.e. it is the memory resident portion of the operating system.

The functions incorporated into the kernel are shown in Figure 10.5. There are three system processes directly under the control of the kernel, these are a command analyser, a high level resource scheduler and a priority controller.

Figure 10.5

The interaction between the operating system and the hardware and software components of a computer.



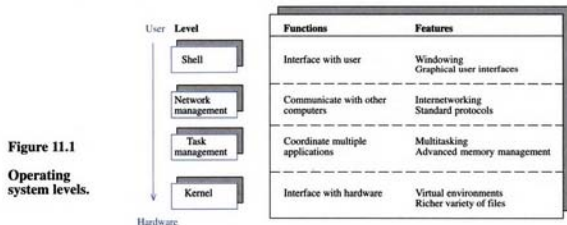
## Chapter 11

### The Operating System (Part 2)

The previous chapter introduced a number of terms used in describing operating systems. This chapter will expand on the initial introduction by considering the main functions in more detail.

#### 11.1 OPERATING SYSTEM LEVELS

Figure 11.1 presents a block diagram of the different levels within an operating system noting the functions and features of the levels. It can be seen that in moving down through the **operating system levels** we are moving from user oriented functions, such as providing information through a graphical interface, to hardware oriented functions, such as controlling the disks to manipulate data and program files.



As the operating system has evolved with the computer, the software for it has tended to grow in layers of procedures. The different software routines within an operating system are often shown diagrammatically as a series of shells, like the different layers within an onion. These layers relate to the different types of functions and services provided by the system. This is shown in Figure 11.2 with an example of the XINU operating system. This operating system was originally developed for a distributed computer network based on DEC LSI 11/02 computers.

Each of the layers in an operating system handles specific features of the system, with the lower layers tending to handle the more basic tasks. The number of layers in an operating system depends on the level of control of the system and the hardware complexity of the processing structures. For example, the operating system for a large mainframe configuration will be more complex and therefore require more layers of software than for a small minicomputer.

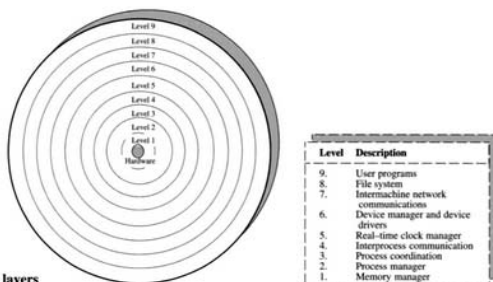


Figure 11.2

The different layers of software within a complex operating system.

The ability to decompose the functions of an operating system into a strict hierarchical structure as shown in Figure 11.2 has a number of advantages for the development of an operating system because of the possibility of developing the levels independently. With structured programming techniques the operating system designer can take reasonable steps to verify that each level of the operating system meets its specifications and that the need for a procedure at a higher level to invoke a lower level service does not introduce unacceptable delays in the progress of the higher level procedure. It is also normal for layered systems to be constructed such that the time required to perform functions at each level is smaller as one conceptually moves closer to the hardware level.

The layering of procedures may be seen as a form of functional hierarchy where a procedure at one level may directly call any visible operation of a lower level. However it must be noted that the level structure shown in Figure 11.2 is only a diagrammatic representation of the procedures within the overall operating system and it does not mean that information must be passed down through all the software layers to get to a lower level. In fact with this form of functional hierarchy no information need flow through any intermediate level but instead each level is able to locate local objects by their internal names without having to rely on some kind of central controlling program.

In Figure 11.2 the outer layers typically have to call the procedures in the inner layers to help them fulfil their functions. At the centre of the layers is the **kernel** of the operating system which will typically provide the following routines.

**1. Process Management:** The kernel controls the overall scheduling of the computer's operations. It pulls required programs from the resident storage devices, e.g. floppy or hard disks, and loads them into primary memory, keeping a list of the jobs run on the processing unit. It handles process scheduling, process manipulation, process control block manipulation and process synchronization.

Operating  
system  
levels

*Kernel*

**2. Communication:** The kernel is responsible for communicating with the system operator. The first type of call to the operating system is usually through the computer operator, typing a request through the system console, which the command interpreter within the operating system interprets, and subsequently causes the operating system routines to perform some series of actions. Within any computing system there is also a certain amount of communication from the operating system to the operator, e.g. messages indicating various events such as error conditions and input/output (I/O) devices that need attention. By processing faults through this central control in an orderly way it is possible to reduce the time lost due to an error in a particular part of the system. This communication is usually controlled by some form of scheduler which routes all system messages to the operator, usually printing them on a printer or on the visual display unit of the system console. In addition the kernel handle interprocess communication, ie. passing messages between processes or tasks.

**3. Interrupt handling:** The kernel is responsible for handling interrupts from either a single or multiuser situation, depending on the system.

**4. Resource Management:** The kernel handles I/O and memory management, support of I/O activities, storage allocation and deallocation and support of the [file system](#).

Glossary 10.1 describes some of the terms used when discussing operating systems. The three major functions of the operating system are process scheduling, memory management and I/O management.

## 11.2 PROCESS SCHEDULING

Scheduling within a computer system is the organization of the sequence in which tasks are run on the system. It is also used to refer to the allocation of resources to these tasks. The scheduler is the mechanism within the operating system that controls the access of the different programs to the central processing unit (CPU). It can essentially be split into the high level scheduler and the low level scheduler.

### 11.2.1 HIGH LEVEL SCHEDULER

The high level scheduler is often called the job or task manager. It is the collection of software procedures within the operating system that controls the movement of processing jobs from the time that they enter the system, usually from a disk device, until they complete, usually with a printout. The high level scheduler governs the policies of resource allocation and influences the order in which tasks are selected for the processor. Some of the major functions of the high level scheduler are as follows.

**1. Task input:** In a time shared computer system the high level scheduler must read the tasks and jobs from the many users and place them on the job queue for access to the CPU.

**2. Scheduling tasks:** Before a job can gain access to the CPU, the high level scheduler must request information from the memory and I/O management routines to ensure that all the resources needed by the job are available. If all the resources are available, the job is then placed in a wait state in the central memory by the memory manager and attached to the service request list of the low level scheduler; the job is now ready to be processed by the CPU. If all the resources needed by the job are not ready, the high level scheduler will select the next highest priority job for scheduling and repeat its analysis of the job's requirements. It then schedules jobs in the input queue by allocating the data sets, main storage space, and