Operating Systems

Sibsankar Haldar
Motorola, Inc.

and

Alex A. Aravind
University of Northern British Columbia

PEARSON

Delhi • Chennai • Chandigarh
Upper Saddle River, NJ • Boston • London
Sydney • Singapore • Hong Kong • Toronto • Tokyo

1. Microsoft Windows (Computer file) 2. Linux. 3. Operating systems (Computers)
I. Aravind, Alex A. (Alex Alagarsamy), 1965- II. Title.
QA76.76.O63H3433 2009
005.4'3 dc22

2008051401

ISBN 978-81-317-1548-2

Head Office: 7th Floor, Knowledge Boulevard, A-8(A), Sector-62, Noida
Registered Office: 14 Local Shopping Centre, Panchsheel Park, New Delhi 110 017, India

Typeset in 10/12 Times Roman by ACEPRO India Private Ltd.
Printed in India at Sanat Printers

Pearson Education Inc., Upper Saddle River, NJ
Pearson Education Ltd., London
Pearson Education Australia Pty, Limited, Sydney
Pearson Education Singapore, Pte. Ltd
Pearson Education North Asia Ltd, Hong Kong
Pearson Education Canada, Ltd., Toronto
Pearson Educacion de Mexico, S.A. de C.V.
Pearson Education-Japan, Tokyo
Pearson Education Malaysia, Pte. Ltd.

Threads

Learning Objectives

After reading this chapter, you should be able to:

- Describe the purpose of a process.
- Define the attributes of a process.
- Distinguish between a process and a program.
- Identify the relationship between a thread and a process.
- Describe process and thread management.

4.1 Introduction

We know from Chapter 2 that a hardware platform is composed of many different hardware resources that do purposeful work. However, general users of a computer may have no interest in knowing about these hardware resources; their primary interest is in running utility programs of the system, the application programs they themselves develop, and the output their programs produce. When a user executes a program, the execution requires various resources to accomplish the task. For example, it needs the CPU and the main memory. We know that the operating system is the sole manager of all resources hardware and software in a computer system and is responsible for their allocation and deallocation. The key question is to whom does the operating system allocate resources? The short answer is, to program executions. We discuss the long answer at length below.

You may have noted that many users may run a single application program simultaneously. A single user may also run an application more than once simultaneously. In short, simultaneous executions of the same program are possible in a computer system. For example, many users may run a particular text editor program (say, vi) simultaneously. Even in such a case, the resource requirements for one editor execution differ from those for other executions of the same editor program.[1]

[1] In this context, program means a standalone executable application program or utility.

they play a vital role in the design, development, and implementation of most operating systems.

most fundamental concept in the context of operating systems, an to understand this concept to comprehend clearly the working modern operating systems.

We frequently use the terminologies program execution, co process interchangeably to mean the same action. Processes and cutions are in one-to-one correspondence with each other. The op starts an execution of a program (i.e., a new computation) process. The process is allocated some main memory to hold it data. When a process is started, the operating system brings (i required program and data in the allocated main memory, and buil "execution context" of the process. Note that each process sta defined initial context. The operating system then allocates a various resources to and from the process as the execution o evolves. The process execution context stores this resource-allo tion, and the current state of the program execution and the oth related to it. In short, the operating system uses a process as a han one-program execution.

When a program is given to the operating system for e system builds the other components of the program execu necessary attributes, and eventually shapes them all into a w that we call a process that can be conveniently, effectively, managed in the operating system to accomplish the program's This brings us to a set of basic questions about processes: wh process? What are its main components and what exactly are play during the lifetime of the process? What are the states that transit through and what kind of privilege modes can it adopt conditions during its lifetime? How are processes created, destroyed in a system?

≫ A process may be viewed as the eventual *avatar* of a solution to a problem. A solution is first abstracted as an algorithm in the software design phase, transformed into an application program in the programming phase, and finally created as a process to solve the intended pro blem by execution of the corresponding program under the premise of an operating system. Every process has a program as its component and, at any given time, it is in a state of executing the program.

This chapter aims to answer the above and related questi processes. The next three chapters also deal with topics prima processes and their management.

4.2 Process Abstraction

Performing a task using a computer essentially requires executing gram. Therefore, an operating system, on receiving a specific tas have a suitable program available to execute the task. The entity w the task described in a given program is called a *process*. (See B evolution of the process terminology.) Every process has a p execution component and acquires its behaviour mostly from th program. In short, program executions are abstracted as processes process executes a program it may also require other information for the execution.

on completio

4.1 Evolution of Process

Historically, large software systems underwent many painful development and maintenance cycles. Many of them failed miserably. Then came the concept of the structured system. The concept of the process was one of the many structuring tools devised to muster the complexity of large software systems. An operating system too is a large complex system, and the process concept has been in use as a structuring tool here as well, especially to handle its runtime complexity. Since its inception, the concept of the operating system has evolved constantly, and has undergone several revisions and advancements. The concept of the process evolved alongside the operating system. (The original terminology employed to describe it was "task", and later "job". The term "process" has superseded them.)

To understand the evolution of the concept of process, we start with the open-shop and closed-shop era. In that era, a computer system had only one CPU available. The whole of the memory and the CPU were given over to one program until it completed the execution; no user interaction was allowed during the execution of that program. In such situation, no additional information was needed to execute the program, and therefore, in the open-shop and closed-shop (batch) systems, a process simply meant a "program or job in execution".

The situation changed in the era of multiprogramming, where more than one program was allowed to share the main memory, the CPU, and other system resources simultaneously. Apart from this, it became possible for many users to request the system to execute the same program simultaneously. Therefore, each program execution has certain fundamental re These requirements are maintaining (owner id, program id, etc.) for its id an exclusive and secure space for its and of other related information (add a mechanism to record its current exe (as it could lose the CPU any tim execution), a stack (to hold values of temporary variables, return addresse operating system specifically suppli as additional information is not ex vided by the given program. This system to effectively execute and program execution even if there are executions of the same and other pro additional information provided by t system and the given program toge logical structure called the process gram execution.

When a process does not have c CPU, it may be waiting for the C from an I/O device, or for data f process in the system. That is, in th process can exist in various logical s its lifetime. This state information a the part of the process. Thus in programming era, a process is to be program execution, along with its management information, and exis state of action. There is no perfect accepted definition of a process. usage of the term seems to have coine MULTICS days in 1960s.

The concept of a thread as an activ process is the latest addition to thi Threads are strands of program execu ded in a process.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | sales@docketalarm.com | 1-866-77-FASTCASE

fastcase®
Smarter legal research.