

Chapter 13

UNIX Operating System

At a Glance

Instructor's Manual Table of Contents

- Overview
- Objectives
- Teaching Tips
- Quick Quizzes
- Class Discussion Topics
- Additional Projects
- Additional Resources
- Key Terms

Lecture Notes

Overview

Unlike many operating systems, UNIX is not limited to specific computers using a particular microprocessor as a CPU. Instead, UNIX systems run on all sizes of computers using a wide range of microprocessors. In addition, current versions of many other operating systems have been revised to include the capability to run UNIX applications and connect smoothly with UNIX networks.

This chapter begins by presenting the evolution of UNIX operating systems. It then discusses design goals. The roles of the Memory, Device, Processor, and File Managers in UNIX systems are discussed. The chapter concludes with a discussion on the user interface in the UNIX operating system.

Learning Objectives

After completing this chapter, the student should be able to describe:

- The goals of UNIX designers
- The significance of using files to manipulate devices
- The strengths and weaknesses of having competing versions of UNIX
- The advantages of command-driven user interfaces
- The roles of the Memory, Processor, Device, and File Managers in UNIX

Teaching Tips

Overview

1. Provide students with an overview of the UNIX operating system. Discuss the three major advantages: portability, powerful utilities, and device independence.
2. Point out the disadvantages of the UNIX operating system. Be sure to mention that there is no single, standardized version of this operating system and that the commands may seem cryptic to novice users.

Teaching Tip

Refer to the following Web site for an overview of UNIX operating systems:
www.bell-labs.com/history/unix/tutorial.html

Teaching Tip	Note: UNIX is case sensitive and strongly oriented toward lowercase characters, which are faster and easier to type. Remind students that throughout this chapter, all filenames and commands are shown in lowercase.
---------------------	---

History

1. Provide students with a brief overview of the history of the UNIX operating system. Use Table 13.1 on page 403 as a guide.

The Evolution of UNIX

1. Explain the evolution of the various UNIX operating systems. Include in your discussion the historical roots of UNIX, its features, and modifications.
2. Discuss the features of newer, more current releases of UNIX. Examples include full support for local area networks, compliance with international operating system standards, improved system security, and the Common Desktop Environment (CDE).
3. Discuss the challenges involved in the development of UNIX, including standardization efforts to improve portability of programs from one system to another. Point out that the release of ISO/IEC 9945:2003 is a major step in this direction.

Teaching Tip	Refer to the following Web sites for information about the history of UNIX: www.theopengroup.org/ www.bell-labs.com/history/unix/
---------------------	---

Design Goals

1. Discuss the design goals of UNIX and list some original goals of the designers of UNIX. These include the immediate goals of developing an operating system that supports software development and uses as simple algorithms as possible, and the long-term goal of making it portable.
2. Point out that popular versions of UNIX meet the additional design element of conforming to the specifications for POSIX.

Memory Management

1. Point out that UNIX was originally designed for single users, but that it eventually developed into powerful operating systems for multiuser environments as well.
2. Explain memory management techniques used in UNIX operating systems, namely swapping and demand paging. Discuss their key features, situations in which they are used, and their pros and cons.
3. Explain the typical internal memory layout for a single user-memory part UNIX image. Discuss the key features, including the purpose and location of each part (program code, data segment, and stack). Use Figure 13.2 on page 407 as a guide and pictorial example.
4. Explain the key features of the UNIX kernel. Point out that UNIX uses the same memory management concepts for network PCs, single-user, and multi-user systems.

Teaching Tip	Refer to the following Web site for a quick view of memory management in UNIX: http://dataexpedition.com/~sbnoble/Tips/memory.html
---------------------	--

Process Management

1. Provide students with an overview of process management in UNIX operating systems. Outline the tasks performed by the Process Manager of the UNIX system kernel.
2. Explain the process scheduling algorithm in UNIX operating systems. Make sure students understand the concept of compute-to-total-time ratio.

Process Table Versus User Table

1. Provide students with an overview of the process control structure in UNIX, using Figure 13.3 on page 410 as a guide.
2. Explain the three tables that UNIX uses to keep the system running smoothly: the process table, user table, and text table.
3. Explain how the process table and text table interact for processes with sharable code, as well as for those without sharable code.

Synchronization

1. Provide students with a brief overview of process synchronization in UNIX operating systems. Discuss how this is achieved using an example such as that provided on page 411 in the text. Explain the situation where a race can occur.

2. Explain the usefulness of *fork*, *wait*, and *exec* commands in process management, using Figures 13.4, 13.5, and 13.6 on pages 412-413 in the text as a guide.

**Teaching
Tip**

System calls such as the `ls` command illustrate the flexibility of UNIX that programmers find extremely useful.

Quick Quiz 1

1. Which of the following truly describe the stack segment in the internal memory layout for a single user-memory part UNIX image? (Choose all that apply.)
 - a. Starts at the highest memory address and grows downward
 - b. Starts after the program code and grows toward higher memory locations
 - c. Section where process information is saved when a process is interrupted
 - d. Sharable section of memory

Answer: a and c

2. UNIX uses the _____ page replacement algorithm.

- a. MRU
- b. LRU
- c. FCFS
- d. FIFO

Answer: b

3. Which of the following commands in UNIX gives the user the capability of executing one program from another program?

- a. `nohup`
- b. `nice`
- c. `fork`
- d. `exexv`

Answer: c

Device Management

1. Discuss the important features of device management in UNIX operating systems.
2. Discuss the incorporation of device drivers into the kernel.
3. Explain the roles of the `config.c` file.

Device Classifications

1. List the two categories into which UNIX divides the I/O system, namely the “block I/O” system (sometimes called the “structured I/O” system) and the “character I/O” system (sometimes called the “unstructured I/O” system).

2. Explain the hierarchy of I/O devices in UNIX using Figure 13.7 on page 415 as a guide. Discuss the roles of the major device number, minor device number, and block I/O system.
3. Explain the mechanisms of character devices.

Device Drivers

1. Provide students with a brief overview of device drivers in UNIX. Point out that although device files may be kept anywhere on the file system, by default and convention they are kept in the /dev directory.

Teaching Tip	An innovative feature of UNIX is its treatment of devices; the operating system is truly device independent. It achieves this independence by treating each I/O device as a special type of file.
---------------------	---

File Management

1. Explain briefly the three types of files in UNIX: directories, ordinary files, and special files.
2. Explain how files are stored in UNIX operating systems.
3. Explain the organization of disks in the UNIX file management system. Discuss briefly the four basic regions into which the disk is divided. Mention that whenever possible, files are stored in contiguous empty blocks.
4. Explain “i-node” and the information it contains on a specific file.

File Naming Conventions

1. Discuss the general rules for filenames in UNIX. Be sure to mention that filenames in UNIX are case sensitive.
2. Describe the hierarchical tree file structure supported in UNIX, using an example as shown in Figure 13.8 on page 418 as a guide.
3. Explain the rules that apply to all path names using the bullet points on page 419 as a guide.

Directory Listings

1. Describe the eight pieces of information for each file that a “long listing” of files in a directory shows, using Table 13.2 on page 419 as a guide. Explain the meaning of each of the ten characters used in the first column in the directory listing.

2. Explain the meaning of each of the characters used in the second column in the directory listing
3. Clarify the concept of “aliases.”
4. Explain the meaning of each of the characters used in the remaining columns in the directory listing. Columns three through five show the name of the group, the name of the owner, and the file size in bytes. Columns six and seven show the date and time of the last modification. The last column lists the filename.

Data Structures

1. Describe the data structure in UNIX, using Figure 13.9 on page 421 as a guide. Point out that the information presented in the directory is not all kept in the same location. UNIX divides the file descriptors into parts, with the hierarchical directories containing only the name of the file and the “i-number,” which is a pointer to another location, the “i-node,” where the rest of the information is kept.
2. Discuss what happens when a file is opened, created, linked, or deleted.

User Interface

1. Remind students that UNIX is a command-driven system. Explain the key features of user commands: they are very short, either one character or a group of characters (an acronym of the words, which make up the command); they cannot be abbreviated or spelled out; and they must be in the correct case. Note that the system prompt is very economical – only one character, (\$) or (%) – and that brief error messages are displayed.
2. Explain various user commands, using Table 13.3 on page 422 as a guide. Explain the general syntax of commands. Note that information in the figure displayed in the accompanying PowerPoint slide is incomplete, and refer the students to pages 422-423 for the complete list of sample commands.
3. Describe the general command syntax and the three components (command, arguments, and filename). Use Figure 13.10 on page 423 in the text to demonstrate the command interface for an MAC OS x system.

Script Files

1. Provide students with an overview of script files. Provide an example as included in the text on page 424 to illustrate their use.

Redirection

1. Explain the redirection command, which is used to redirect output from one standard input or output device to another. Use examples to illustrate. Point out that redirection works in the opposite manner as well. Emphasize the concepts of reverse direction and direction. Discuss the “append” command as well.
2. Point out that redirection can be combined with system commands to achieve results not possible otherwise. Provide examples of the use of the redirection command in combination with “who” and “sort” commands as provided on page 425.
3. Be sure to emphasize that the interpretation of < and > is done by the shell and not by the individual program.

Pipes

1. Explain the use of the pipe command with examples. Point out that it is possible to combine pipes and other filters.
2. Discuss pipeline using an example such as that provided on page 426.

Filters

1. Explain various filter commands, such as *wc* and *sort*. Provide different examples to illustrate the use of the sort command to arrange files in ascending order (numerically or alphabetically), in reverse order, or to sort the files by column.

Teaching Tip	Refer to the following Web site for more information on UNIX Pipes and redirection: http://polishlinux.org/console/unix-pipes-streams-and-redirections-explained/
---------------------	--

Additional Commands

1. Illustrate, using examples, some additional commands that are often used in UNIX such as *man*, *grep*, *nohup*, and *nice*. Point out that *nohup* automatically activates *nice* by lowering the process’s priority; however, the opposite is not true.

Teaching Tip	Refer to the following Web site for a detailed list of UNIX/Linux commands including their syntax and use: www.computerhope.com/unix/overview.htm
---------------------	---

Quick Quiz 2

1. The _____ is used as an index to the array to access the appropriate code for a specific device driver.
 - a. i-number
 - b. Minor device number
 - c. Major device number
 - d. Stack number

Answer: c

2. If you want to put a very long job on the background, work on some other jobs, and log out before the long job is finished, which of the following commands will you use?
 - a. nohup
 - b. nice
 - c. fork
 - d. exexv

Answer: a

3. Recent versions of UNIX have a program called config that will automatically create a(n) _____ file for any given hardware configuration.

Answer: conf.c

Class Discussion Topics

1. Have students discuss device management in the UNIX operating systems. How is it different from device management in the Windows operating systems? Ask students to list all the advanced key features of device management in UNIX operating systems.
2. Ask students if any of them have worked with UNIX in the past, and if so, how they felt about the experience. What were some of the challenges that they faced in attempting to install and configure UNIX? Do they think that UNIX will be a long-term and viable alternative to Windows? Why or why not?

Additional Projects

1. Have students research online and compile a chart illustrating the major advantages and disadvantages of the Windows and UNIX operating systems. The chart should be a guide for users who may consider making a change from Windows to UNIX or from UNIX to Windows.
2. Have students research online and create a list of what they consider to be the best online UNIX resources available, along with the types of information each site provides. Once completed, compile the students' lists into a master list to be distributed to the class.

Additional Resources

1. IBM.com:
www.ibm.com
2. IBM's UNIX servers and AIX documentation:
www-1.ibm.com/servers/eserver/pseries/
3. Sun.com:
www.sun.com/
4. IEEE:
www.ieee.org
5. The Open Source:
www.theopengroup.org/

Key Terms

- **Argument:** in a command-driven operating system, a value or option placed in the command that modifies how the command is to be carried out.
- **Child process:** in UNIX-like operating systems, the subordinate processes that are created and controlled by a parent process.
- **Command line interface:** an interface that accepts typed commands, one line at a time, from the user. It contrasts with a graphical user interface.
- **Compaction:** the process of collecting fragments of available memory space into contiguous blocks by moving programs and data in a computer's memory or secondary storage.
- **CPU-bound:** a job that will perform a great deal of nonstop processing before issuing an interrupt. A CPU-bound job can tie up the CPU for long periods of time.
- **Device driver:** a device-specific program module that handles the interrupts and controls a particular type of device.
- **Device independent:** programs that can work on a variety of computers and with a variety of devices.
- **Directory:** a logical storage unit that contains files.
- **I/O-bound:** a job that requires a large number of input/output operations, resulting in much free time for the CPU.
- **Kernel:** the part of the operating system that resides in main memory at all times and performs the most essential tasks, such as managing memory and handling disk input and output.
- **Multitasking:** a synonym for multiprogramming, a technique that allows a single processor to process several programs residing simultaneously in main memory and interleaving their execution by overlapping I/O requests with CPU requests.
- **Parent process:** in UNIX-like operating systems, a job that controls one or more child processes, which it created.

- **Portable Operating System Interface for Computer Environments (POSIX):** a set of IEEE standards that defines the standard user and programming interfaces for operating systems so developers can port programs from one operating system to another.
- **Race:** a synchronization problem between two processes vying for the same resource. In some cases, it may result in data corruption because the order in which the processes will finish executing cannot be controlled.
- **Reentrant code:** code that can be used by two or more processes at the same time; each shares the same copy of the executable code but has separate data areas.
- **Script:** a series of executable commands written in plain text that can be executed by the operating system in sequence as a single procedure.
- **Sharable code:** executable code in the operating system that can be shared by several processes.