HP Computer/Instrument Systems
Training Course

UNIX System Basics I

Instructor Guide

HEWLETT
PACKARD

51489-90011
Printed in U.S.A. 12/91

Notice

Product Development
Professional Services Division
100 Mayfield Avenue
Mountain View, CA 94043 U.S.A

Contents

Contents

Contents

Contents

Contents

Contents

Overview

Course Description

This course is designed to introduce the fundamentals of using a computer running the UNIX® operating system. It assumes that the student is not very familiar with computers or has had little computer experience. The class introduces, reinforces, and tests all of the basic concepts necessary for an individual to become an effective user of a UNIX operating system computer. There are also several advanced topics presented as advanced exercises and optional modules.

Course Goal

To prepare the student to use applications on a computer using the UNIX (or a similar) operating system.

Student Performance Objectives

Upon completion of this course, the student will be able to:

- Explain what an operating system is.
- Explain what a UNIX operating system is.
- Explain the relationship of HP-UX to UNIX®.
- Identify the hardware for a basic UNIX system.
- Explain the function of the UNIX system console.
- Explain the functions of a terminal's special typing keys.
- Explain the functions of a terminal's editing keys.
- Log in to and log off of a UNIX system.
- Explain the need for and be able to use the erase and line kill characters.
- Execute commands with and without arguments.
- Change your password.
- Use the `date` and `who` commands to obtain system information.
- Use the `mailx` and `news` commands for communicating with system users.
- Use the `echo` and `banner` commands for displaying text.
- Use the *HP-UX Reference Manual* to explain the purpose of each section.
- Access the online copy of the *HP-UX Reference Manual.*
- Understand the UNIX system file tree structure.
- Explain the role of directories.
- Explain the difference between an absolute path name and a relative path name.

Overview

- Use the `ls`, `cd`, and `pwd` commands to traverse the directory structure.
- Understand how to organize related files.
- List the names of currently existing files.
- Examine the contents of a file.
- Copy an existing file to a file of another name.
- Rename existing files.
- Remove existing files.
- Explain the meaning of file permissions.
- Change file and directory permissions.
- Describe and change the owner and group attributes of a file.
- Create and remove directories.
- Use basic file name generating characters.
- Explain the three modes of `vi`.
- Enter and exit the `vi` editor.
- Use the cursor movement keys in `vi`.
- Use the `vi` editor to create and store a new text file.
- Use the appropriate `vi` commands to insert text.
- Use the appropriate `vi` commands to delete text.
- Use the appropriate `vi` commands to modify text.
- Use the `vi` editor to modify an existing text file.
- Describe what the K-shell is.
- Explain the features of the K-shell.
- Use the K-shell command line recall and editing features.
- Use file name completion.
- Use the K-shell alias feature.
- Set up your environment to customize the K-shell.
- Explain the concept of input and output redirection and pipelines.
- Explain the meaning of stdin and stdout.
- Use output redirection to create files.
- Use pipes to connect commands.
- Explain the purpose of the line printer spooler system.
- Identify and use the line printer spooler commands used to interact with the system.
- Monitor the status of the line printer spooler system.
- Describe what happens when someone logs in.

Overview

- Describe the user environment variables and their functions.
- Understand and change specific environment variables such as *PATH* and *TERM*.
- Customize the user environment to fit a particular application.
- Describe the system boot procedure.
- Check what is running on the system.
- Kill unwanted processes.
- Decribe the system shutdown guidelines.
- Cleanly shut down the system using the `shutdown` command.
- Use Key Shell softkeys to perform HP-UX commands.
- Translate a softkey command into HP-UX syntax.
- Use command line editing to correct mistakes.
- Use online help for pre-configured softkeys and selected topics.
- Explain what HP VUE is.
- Use the mouse to control the HP VUE environment.
- Use the HP VUE Workspace Manager.
- Work with terminal windows.
- Use the File Manager to manipulate files and directories.
- Use the Help Manager to obtain information on a topic.
- Use the Style Manager to customize the HP VUE environment.
- Describe the different network services in HP-UX.
- Explain the function of a Local Area Network (LAN).
- Find the host name of the local system and other systems in the LAN.
- Use the ARPA/Berkeley Services to perform remote logins, remote file transfers, and remote command execution.

Reference Documentation

These are a few books that come with your system that you may find useful when you start learning HP-UX.

- *HP-UX Reference Manual*—Section 1 (part number B1864-90000)
- *A Beginner's Guide to HP-UX* (part number B1862- 90000)
- Optional: *Shells: User's Guide* (part number B1862-90017)
- Optional: *The Ultimate Guide to the vi and ex Text Editors* (part number 97005-90015)
- Optional: *HP Visual User Environment User's Guide* (part number B1171-90022)

Overview

Agenda

This is the basic schedule of the class. This schedule will vary depending on the number of questions raised or the level of interest shown by the students. Appendixes A-D are optional and present extra material that may be covered if the majority of the class has an interest (time permitting, of course). The appendixes may be presented at any time.

It will be a busy two days. Just relax and take the course one module at a time. The material and exercises are designed to reinforce topics you have previously learned and to introduce new topics. Most of all, have fun!

Day 1	AM	Module 1—Introduction
		Module 2—Computer Basics
		Module 3—Logging in and Orientation
	PM	Module 4—Directories and Files
		Module 5—Managing Files
		Module 6—More on Files and Directories
Day 2	AM	Module 7—The vi Editor
		Module 8—The Korn Shell
	PM	Module 9—Changing Input and Output
		Module 10—Using the Line Printer
		Module 11—The User Environment
		Appendix A—Running your HP-UX System
		Appendix B—Using the System with Key Shell
		Appendix C—HP Visual User Environment (VUE)
		Appendix D—Using Network Services
		Appendix E—Standards and Conformance Table
		Appendix F—Lab Notes for MPE Users

Introduction

The instructor kit you have received contains most of the materials you need to teach the *UNIX System Basics I* (51489) course. This document is intended to help you understand the structure of the course materials and hopefully give you some pointers that may help you make the best use of them. Since the courses in the HP-UX curriculum use a different approach than traditional HP training materials, please take some time to read this document carefully.

Overview

Materials List

Instructor Materials

- Instructor kit (part number 51489-60008)
  - Instructor guide (part number 51489-90011)
  - Overhead slide set (part number 51489-80003)
  - Lab tape (part number 51489-10005)

Student Materials

- Student kit (part number 51489-60007)
  - Student workbook (part number 51489-90010)

Student Workbook

The student workbook has been designed to be used both in class and as a reference tool. It is also *the main reference* for the instructor. The student workbook consists of the following parts:

Table of contents	The table of contents lists all of the modules and topics included in the course.
Course	The course is divided into small, manageable modules. Each module is numbered and begins with an objectives page with a list of goals for the student. The pages within the modules are numbered. The slides are always shown above the student notes, which contain a detailed discussion of the material covered in the slides. The instructor should use his or her discretion in deciding how much of this material to cover in class. The actual slide that will be used in class is also on the page.

Instructor Notes

The instructor notes are an extremely useful preparation tool. They are designed to aid the instructor in *preparing* to teach, and, since the instructor notes are a superset of the student notes, they are a great reference.

The notes are a collection of hints that will help the instructor teach the course effectively in a highly interactive manner. The actual course content is in the student workbook. The instructor notes have been developed over a long period of time by people who have taught similar courses. Please take time to review the instructor notes while you are preparing for this course.

Lab Tape and Setup Instructions

The lab tape includes all of the files and directory structures needed for all of the labs in this course. These course lab files are not available anywhere else. The labs are provided on a 150-foot Certified Data Cartridge tape (part number 88140SC). To load the tape do the following:

1. Log in as root.
2. "cd" to some directory; `/tmp` is acceptable.

3. Type: `tcio -i /dev/xxx | cpio -icd`, where *xxx* stands for the device file corresponding to your tape drive.

There are several files on the tape:

- A `README` file that describes the files on the tape and how to install them.
- The `lab.setup` script. This is a program that will set up the files for all of the students so you do not have to do it by hand.
- The `lab.files` directory. This directory contains all of the files needed for the labs in this course.

The following is a description of how to use the `lab.setup` script to install the student files. You need to be superuser to use the program since it attempts to change ownership and permissions on the files.

1. Create entries in `/etc/group` and `/etc/passwd` for all students.
2. Edit the `lab.setup` script. Modify the *student_list* variable to contain the login names of all the students.
3. Type `lab.setup`.

The `lab.setup` script will take the following actions, as necessary, for each student:

1. Create `HOME` directory.
2. Copy `/etc/d.profile` to `$HOME/.profile` (for the Korn shell).
3. Copy the lab files to `$HOME`.
4. Change the ownership, group, and permissions for all files and directories.

The `.profile` file will *not* be overwritten if it already exists.

Classroom Materials

The following manuals should be available in the classroom.

- *HP-UX Reference Manual*—Section 1 (part number B1864-90000) (one copy per two students)
- *A Beginner's Guide to HP-UX* (part number B1862- 90000) (one copy per classroom)
- Optional: *Shells: User's Guide* (part number B1862- 90017) (one copy per classroom)
- Optional: *The Ultimate Guide to the vi and ex Text Editors* (part number 97005-90015) (one copy per classroom)
- Optional: *HP Visual User Environment User's Guide* (part number B1171-90022) (one copy per classroom)

Ordering Information for the Master Index Manual

There is a *Master Index* Manual (part number B1862- 90003) that can help your students determine the correct HP-UX manual for the task they need to do. It covers System Administration and Installation, General Usage, HP-UX User's Guides, and General Programming. You may wish to order some copies for your class.

Overview

Equipment Requirements

HP-UX 8.0 or later is required for this course.

Minimum Requirements

- One terminal per two students.
- Any machine capable of supporting the number of terminals required (based on projected student load).
- High Speed Disk Interface card.
- Direct Memory Access (DMA) card.
- Sufficient disk space for HP-UX system software and additional disk space for each user as described in the configuration guide.
- Sufficient random-access memory (RAM) (based on the number of users) as described in the configuration guide.
- A cartridge tape unit for uploading course exercises and sample files. Note that a separate cartridge tape unit is not required if a cartridge tape drive is an integral part of one of the disk drives.
- One printer per system.

Recommended Equipment

One additional terminal for the instructor is recommended.

It is possible to teach this course using a single system (for example, a Series 800) as long as sufficient disk space, RAM, and multiplexer (MUX) ports are provided. If more than one system is used, it is recommended that the systems be interconnected through a local area network (LAN). If Series 300 machines are used, one system per four terminals is a minimum requirement, and one system per three terminals is recommended. Since there are two students per terminal, three Series 300 machines with three terminals each would support 18 students.

The Socratic Method of Teaching

Hewlett-Packard uses an instructional methodology called the **Socratic Method.** This methodology encourages the use of highly interactive training. It is based on the theory that learning is an *active* rather than a *passive* process, and thus active involvement on the student's part will highly improve the quality of the instruction.

Instructor's Guide to the Instructor Notes

Introduction

The teaching techniques covered in the HP-UX curriculum courses emphasize that lecture time must be balanced with class interaction in order to achieve maximal learning. To help the instructor deliver a balanced, interactive class, Hewlett-Packard developed instructor guides for their courses. The instructor guide is divided into the same modules and topics as the student workbook. In fact, the instructor book is the same as the student book with the addition of a set of instructor notes corresponding to each topic.

The instructor guide is intended for HP instructors. It describes the terminology and teaching style so that HP instructors can make the best use of the HP-UX course materials.

Structure of the Instructor Notes

Each module has a module overview that provides general information about the module to be taught, including the motivation for teaching this module at this point in the course. The student workbook does not contain the module overviews. Each module topic includes instructor notes which follow the student notes. The page numbers in the instructor guide do not match the page numbers in the student workbook.

How to Use the Instructor Notes

The purpose of the instructor notes is to help an instructor internalize a new course. They provide suggestions, from one instructor to another, on how to teach the course. The instructor notes save preparation time while allowing the instructor flexibility. Anyone contemplating teaching any of the HP-UX courses should know 250 percent of the subject matter: the instructor notes are not designed to teach the material to a would-be instructor. Rather, the instructor notes give the instructor information that will help him or her teach the course. These are notes that explain motivation, potential problems, and the intended focus of each slide in the course. Ultimately, the person at the front of the class is the boss and does whatever needs to be done. The instructor notes are just good advice.

Definition of Headings in the Instructor Notes

Each student workbook topic has an associated and correspondingly numbered instructor notes page. All instructor notes pages have the following fixed set of headings:

Purpose	Why is this slide here and not somewhere else in the course? How might the students use the information presented on the slide?
Key Points	What is the important concept or skill we are trying to communicate? This is not always applicable, as in the case of an example.
Teaching Question	One or more teaching questions that can help the presentation.
Positive Instance	One or more positive instances for this teaching point.
Negative Instance	One or more negative instances for this teaching point.
Where Problems Arise	Where have classes historically had problems with this teaching point? This category grows as our experience with teaching the course grows.

Evaluating Question	One or more evaluation questions.
Presentation Suggestions	Consists of other suggestions that you can draw upon to enhance the presentation of this slide.
Slide Notes	Points out interesting nuances of the slide that might be missed and also suggests ways of effectively using the slide.
Chalkboard Notes	Suggests when a diagram would be helpful and often suggests what to draw.
Transition	Why does the next slide follow this one? How can I motivate the students to turn their pages?

Course Pace—Read This!

The following is a suggested course pace:

	Morning Modules	Afternoon Modules
Day 1	1-3	4-6
Day 2	7-8	9-11

The following four modules are optional:

- Appendix A—Running Your HP-UX System
- Appendix B—Using the System with Key Shell
- Appendix C—HP Visual User Environment (VUE)
- Appendix D—Using Network Services

You are not expected to teach all four optional modules. They are provided in case the students have a genuine interest in any or all of these topics. The eleven modules and the four optional modules together comprise about a 3-day course. Plan your time according to the pace of the students.

Exercises

In almost every module there are several exercises marked "Advanced." Do not wait for all of the students to finish the advanced exercises. If you do, the course will turn into a 3-day class! These exercises are meant for the more advanced people who work through the normal exercises quickly. Move on when most of the students have finished the regular exercises.

Teaching This Course in One Day

This 2-day course can be scaled down and tailored to be presented in 1 day. Each module is outlined below, along with suggestions on how each can be scaled down for a 1-day course. In general, most of the slides do not need to be presented in as much detail as in the 2-day course. Most of the exercises do not need to be performed by the students. To save a great deal of class time, the exercises can be used as class discussions.

Module 1—Introduction

Retain this module to introduce the UNIX operating system. There are no exercises in this module.

Module 2—Computer Basics

This module may be omitted. Most students would probably already have had some kind of computer exposure or experience.

Module 3—Logging in and Orientation

Retain this module to introduce how to log in to the UNIX Operating System. The exercises in this module duplicate the examples presented on the slides. These exercises are a reinforcement of the examples and may be omitted. The details of the reference manual may also be omitted.

Module 4—Directories and Files

Retain this module to introduce the UNIX operating system file system and how to traverse it. The exercises are a reinforcement of the examples and may be omitted.

Module 5—Managing Files

Retain this module to introduce commands to manage directories. All of the basic file manipulation commands are described. There are many exercises throughout the module to reinforce the topics.

Module 6—More on Files and Directories

This module may be omitted. It is a module full of advanced file manipulation topics.

Module 7—vi Editor

Retain this module to introduce the `vi` text editor. For the 1-day course, this module should be used to introduce the `vi` text editor, not to teach how to use it in depth. The module is broken into sections of commands (for example, insertion commands, deletion, and so on). The module moves more quickly if you work through the slides rapidly and let the students get to the exercises at the end of each section.

Module 8—The Korn Shell

Retain this module to introduce recalling commands and command line editing. The other topics in the module may be omitted.

Module 9—Changing Input and Output

This module may be omitted. It deals with the topics of I/O redirection and pipelines. If the class is going to be administering a system, this should be included.

Module 10—Using the Line Printer

Retain this module to introduce commands to manage the line printer spooler system. It is fairly short and can be done quickly.

Module 11—The User Environment

This module may be omitted if students will not be setting up their own user environments (in other words, if Hewlett-Packard has fully configured their applications). The module is included because most applications require some sort of modification to the user environment.

Appendix A—Running Your HP-UX System

This module may be omitted. It deals with bootup, process status, kill, and shutdown. This is minimally applicable to users of large multiuser systems.

Appendix B—Using the System with Key Shell

This module may be omitted. Key Shell is an alternate user interface available on HP computers. It is very popular with application users.

Appendix C—HP Visual User Environment (VUE)

This module may be omitted. Most students would not use VUE unless they were in an HP workstation environment.

Appendix D—Using Network Services

This module may be omitted. It includes an introduction to the ARPA and Berkeley Services. Basic forms of the common user commands are covered.

The following is a suggested course pace for teaching this course in one day:

	Morning Modules	**Afternoon Modules**
Day 1	1, 3, 4 and 5	7, 8, 9 and 10

Module 1 — Introduction

Objectives

Upon completion of this module, you will be able to do the following:

- Explain what an operating system is.
- Explain what the UNIX® operating system is.
- Explain the relationship of HP-UX to UNIX systems.

Module 1 — Introduction

Overview of Module 1

This module provides basics about what HP-UX is as an operating system as well as an operating "environment."

It should take no more than 30 minutes to cover.

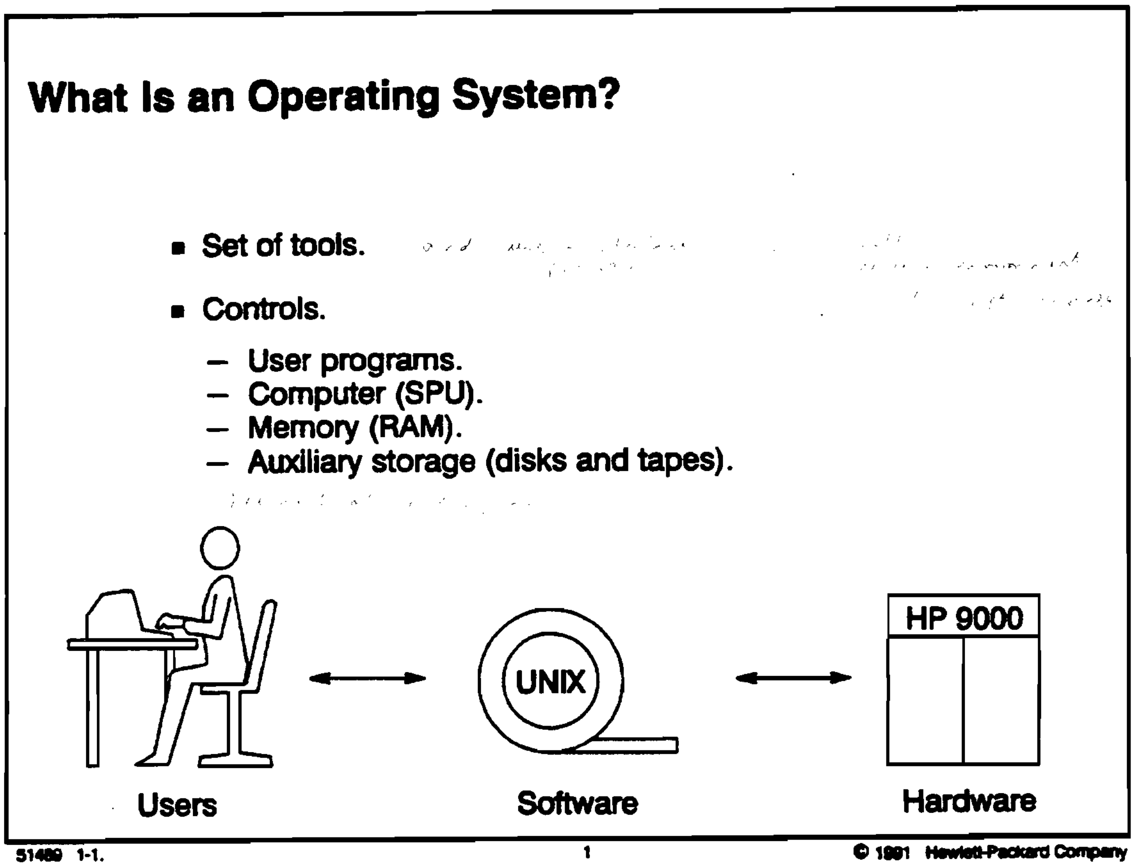
Motivation

This module is intended to clarify for the students what they can expect from the course.

Module Transition

Some basic computer topics are presented next for those who have no computer experience.

1-1. SLIDE: What Is an Operating System?



What Is an Operating System?

- Set of tools.
- Controls.
  - User programs.
  - Computer (SPU).
  - Memory (RAM).
  - Auxiliary storage (disks and tapes).

UNIX
HP 9000
Users
Software
Hardware

51489 1-1. 1 © 1991 Hewlett-Packard Company

Student Notes

People use computers to accomplish particular tasks. These tasks may range from simple text manipulation to complex three-dimensional design work or database interaction. In any case, the user needs to tell the computer what to do. Since few people actually think and use computer instructions, users interface with an **operating system** to get their work done. An operating system is a program (software) that controls the computer (hardware). In the context of this course, an operating system controls one computer running multiple tasks. Almost every computer (from small microcomputers to large mainframes) uses an operating system.

The operating system of any computer must efficiently control all of its resources to provide adequate response to user requests. Some of the computer resources that you should be familiar with include:

- System processing unit (SPU).
- Random-access memory (RAM).
- Auxiliary storage (disks and tapes).

A few of the tasks that operating systems allow the user to do include running programs, printing information, displaying text or graphics on the screen, and reading information from the disk.

UNIX® is an operating system written by AT&T Bell Laboratories that acts as the interface between the users and the computer. It is written in a language called C so it can be used on many different kinds of computers. Besides being an operating system, the UNIX system provides a set of tools and programs to help the user accomplish the needed tasks.

The UNIX system also provides a set of tools called **shells**. Shells are user-interface programs that make the UNIX system easier to use. When you type commands in the UNIX operating system you are really talking to the shell. The shell interprets what you type and passes the information to the UNIX operating system, which, in turn, executes your command on the computer.

HP-UX is Hewlett-Packard's implementation of the UNIX operating system.

1-1. SLIDE: What Is an Operating System?

Purpose

To introduce the highlights of the UNIX operating system environment.

Key Points

- UNIX is a trademark of AT&T.
- The UNIX system is written mostly in the C programming language, which makes it portable to many different makes of computers.
- It is also an operating system with a set of tools (utility programs) and user-interface programs (shells).
- Utilizing the set of tools and the constructs required of any programming language (branching and looping), UNIX operating system shells provide a high-level programming language called the shell programming language, which is very conducive to software prototyping and documentation (courseware) development.
- The UNIX system provides a hierarchical file system that allows the user to establish file naming conventions that are meaningful to the user.
- This course has been written and printed using UNIX system features and facilities.

Make sure you also mention the concept of a shell. It will be important later.

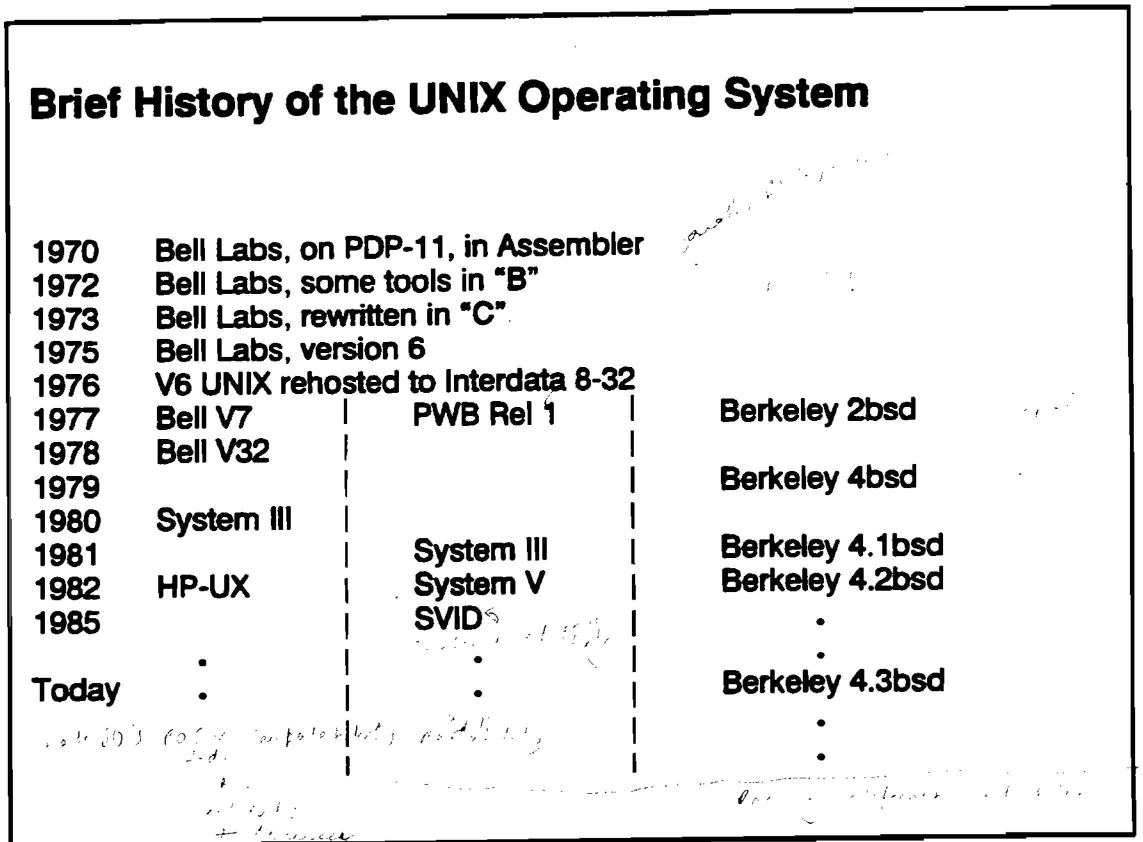
Teaching Question

Ask for examples of operating systems (for example, MS-DOS, CP/M, IBM's MVS, and so on).

Transition

How did the UNIX operating system come about?

1-2. SLIDE: Brief History of the UNIX Operating System



Brief History of the UNIX Operating System

1970	Bell Labs, on PDP-11, in Assembler		
1972	Bell Labs, some tools in "B"		
1973	Bell Labs, rewritten in "C"		
1975	Bell Labs, version 6		
1976	V6 UNIX rehosted to Interdata 8-32		
1977	Bell V7	PWB Rel 1	Berkeley 2bsd
1978	Bell V32		
1979			Berkeley 4bsd
1980	System III		
1981		System III	Berkeley 4.1bsd
1982	HP-UX	System V	Berkeley 4.2bsd
1985		SVID	•
	•	•	•
Today	•	•	Berkeley 4.3bsd
			•
			•

51489 1-2 2 © 1991 Hewlett-Packard Company

Student Notes

The UNIX operating system started at Bell Laboratories in 1969 on a computer that nobody wanted. Ken Thompson, supported by Rudd Canaday, Doug McIlroy, Joe Ossana, and Dennis Ritchie, wrote a small general purpose time-sharing system that started to attract attention. With a promise to provide good document preparation tools to the administrative staff at the Labs, the early developers obtained a larger computer and continued development.

In the early 1970s, the UNIX system was licensed to universities and gained a wide popularity because of the following advantages:

- It was small. (It fit on their minicomputers.)
- It was flexible. (In those days, they gave universities the source code.)
- It was cheap. (They gave it to schools for the price of the tape.)

These advantages offset the following disadvantages of the system that existed at that time:

- It had bugs.
- It had little or no documentation.
- It had no support.

When Thompson introduced the UNIX operating system to the University of California at Berkeley Computer Science Department, it became very popular, and they developed their own version of the system that included many improvements. This version of the UNIX system became known as BSD (Berkeley Software Distribution).

When AT&T realized the potential of the UNIX operating system, they started licensing it commercially. AT&T and Berkeley both continue to make improvements and add more features.

Today, the UNIX operating system is accepted as the operating system standard for multiuser machines.

There are four kinds of UNIX systems:

- AT&T.
- Berkeley (BSD).
- Source-licensed systems (such as HP-UX).
- UNIX system look-alikes.

HP-UX is a source-licensed UNIX system, not merely a look-alike. AT&T requires its "licensees" to call their product something other than UNIX®.

1-2. SLIDE: Brief History of the UNIX Operating System

Instructor Notes

Purpose

Give a brief look at the history of this popular operating system and how HP-UX fits into the UNIX operating system world.

Key Points

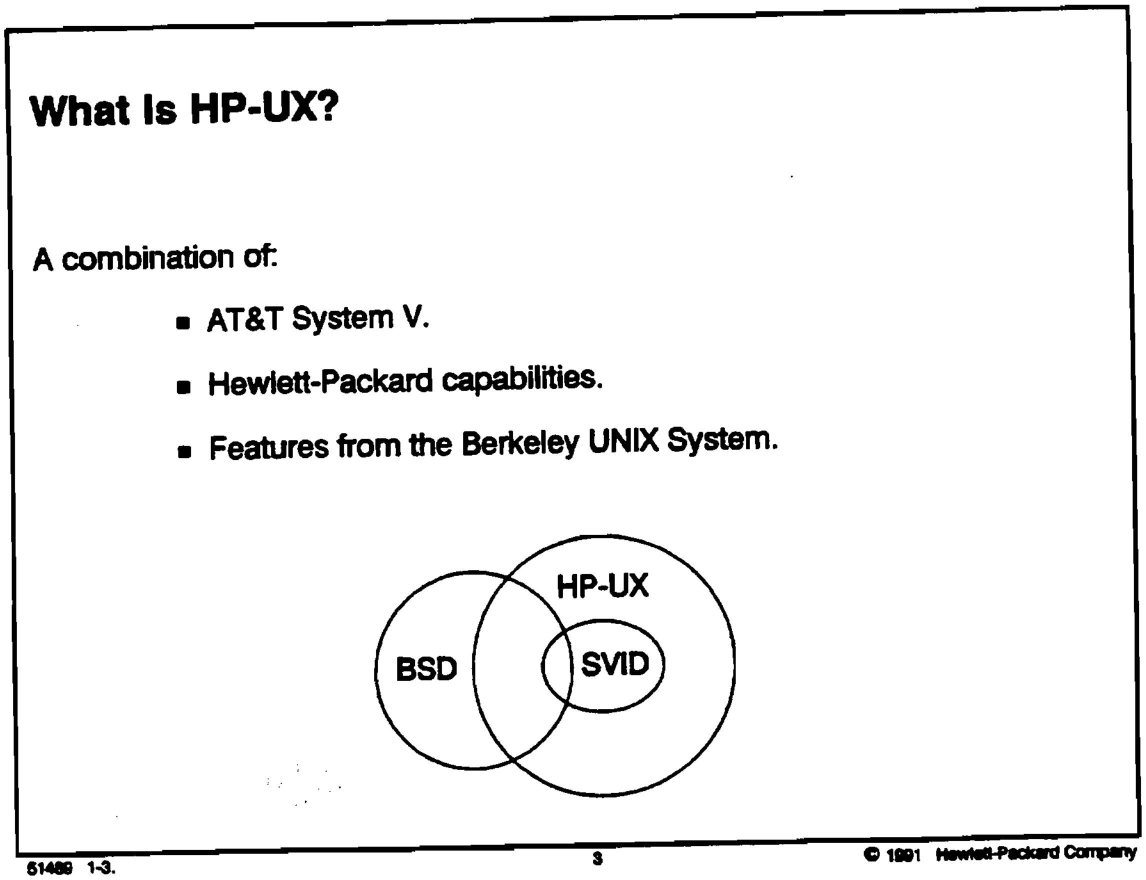
If students are interested in some of the differences between the versions, this information may help. Be careful not to confuse them with the advanced terms. Most of the text is for their information only. Do not spend a lot of time here. Just mention the main points.

From 1970 to 1976, the UNIX operating system stayed within Bell Labs. During this time, the UNIX system resided primarily on the PDP series of computers and could only address 64 kilobytes. These were not virtual memory machines. In 1977 Bell V7 incorporated swapping (entire processes had to reside in physical memory). In 1979, Berkeley 4 incorporated paging (entire processes did not have to reside in physical memory). Today's HP-UX incorporates both paging and swapping algorithms to minimize memory fragmentation and access times. HP-UX is System V.2 and adheres to the System V Interface definition.

Transition

More information about HP-UX follows.

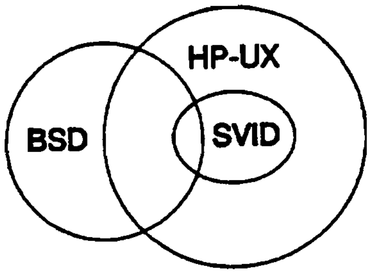
1-3. SLIDE: What Is HP-UX?



What Is HP-UX?

A combination of:

- AT&T System V.
- Hewlett-Packard capabilities.
- Features from the Berkeley UNIX System.



51489 1-3. 3 © 1991 Hewlett-Packard Company

Student Notes

HP-UX is AT&T System V plus "HP value added." HP value added includes both Hewlett-Packard capabilities such as graphics and networking and features from the BSD UNIX system developed at the University of California at Berkeley and elsewhere.

AT&T has established the System V Interface Document (SVID) to describe the standard for AT&T System V compatibility. HP-UX conforms to this standard.

1-3. SLIDE: What Is HP-UX?

Purpose

Students will want to know how this system is different from other UNIX systems.

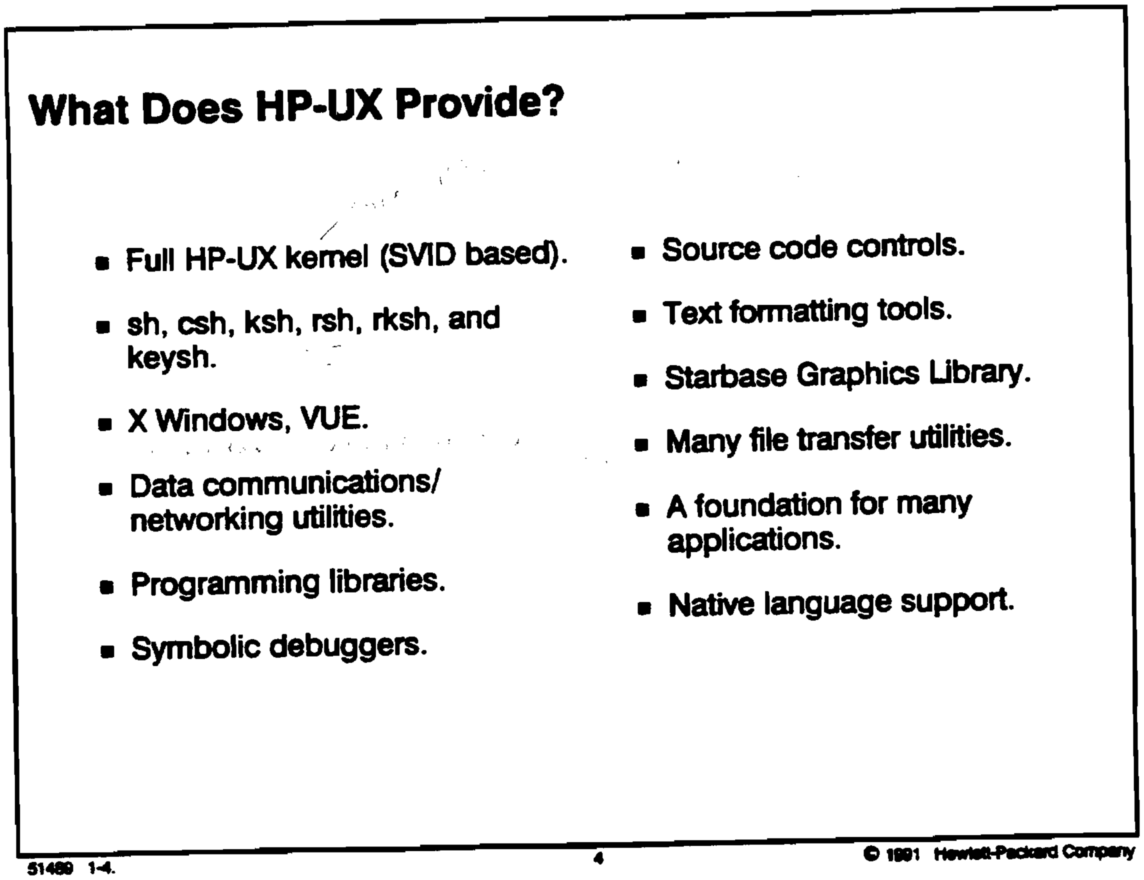
Key Points

HP-UX combines the SVID and Berkeley versions and Hewlett-Packard's own enhancements.

Transition

What does HP-UX give you?

1-4. SLIDE: What Does HP-UX Provide?



What Does HP-UX Provide?

- Full HP-UX kernel (SVID based).
- sh, csh, ksh, rsh, rksh, and keysh.
- X Windows, VUE.
- Data communications/ networking utilities.
- Programming libraries.
- Symbolic debuggers.
- Source code controls.
- Text formatting tools.
- Starbase Graphics Library.
- Many file transfer utilities.
- A foundation for many applications.
- Native language support.

51489 1-4. 4 © 1991 Hewlett-Packard Company

Student Notes

The core of HP-UX is called the **kernel**. This is the actual operating system that controls the computer resources.

The HP-UX system also includes many tools such as shells, networking capabilities, text editors, windowing systems, data communication utilities, and the many other utilities that UNIX system users have come to expect. Many applications and utilities are available for HP-UX.

1-4. SLIDE: What Does HP-UX Provide?

Instructor Notes

Purpose

Students will want to know what HP-UX gives them.

Chalkboard Notes

You may want to discuss the origin of the term **shell**. Draw a series of concentric circles with the inner circle labeled **hardware**, the next one labeled **kernel**, the next labeled **ksh**, **sh**, **csh**, and the last labeled **users**. The students can then see the shell as being similar to an egg shell. This illustration may help them remember the term.

Transition

Let's discuss some computer basics for those of you with no computer experience. :

Module 2 — Computer Basics

Objectives

Upon completion of this module, you will be able to do the following:

- Identify the hardware for a basic UNIX system.
- Explain the function of the UNIX system console.
- Explain the functions of a terminal's special typing keys.
- Explain the functions of a terminal's editing keys.

Module 2 — Computer Basics

Overview of Module 2

This module presents an overview of the hardware of a computer and provides computer basics about UNIX systems.

It should take no more than 30 minutes to cover.

Motivation

Every student must understand some computer basics before learning more detail about the UNIX system.

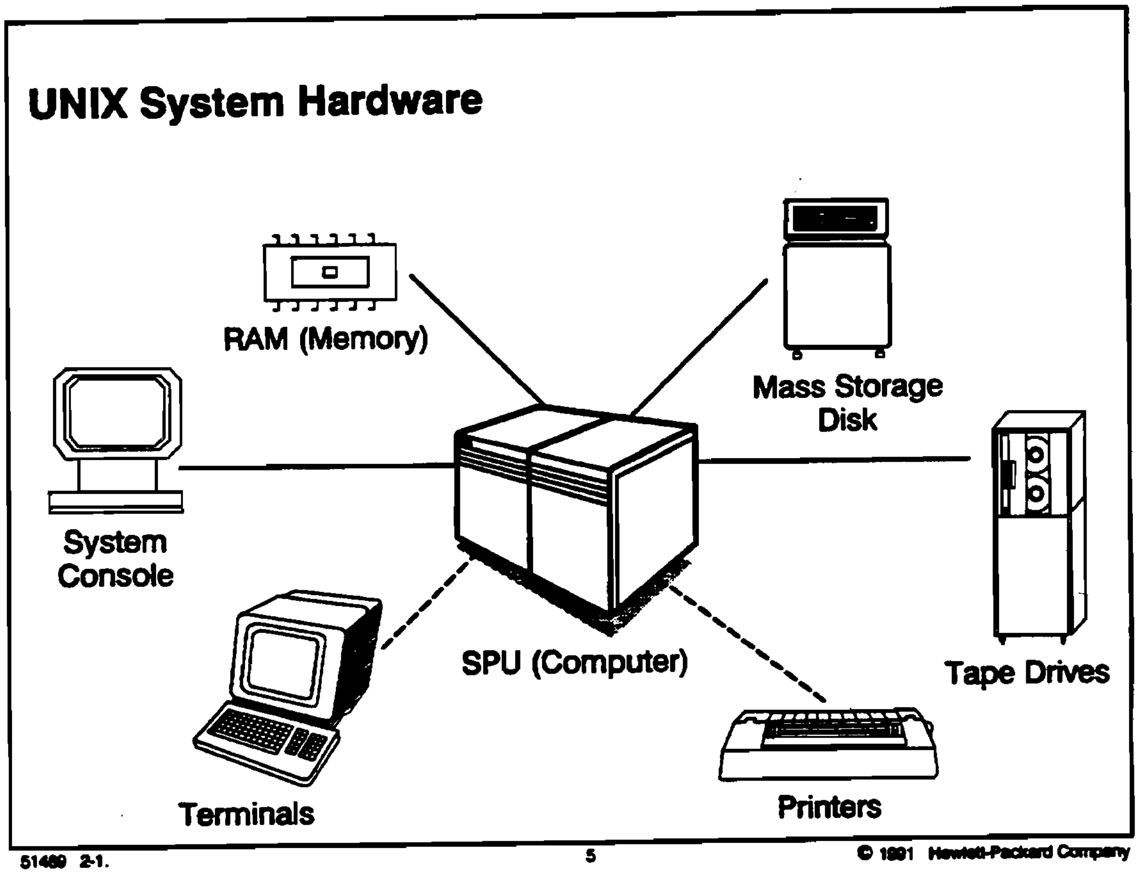
Exercises

The hardware identification exercise is located at the end of this module.

Module Transition

The first thing any user must do to accomplish anything in the UNIX operating system is to log in.

2-1. SLIDE: UNIX System Hardware



UNIX System Hardware

Student Notes

A UNIX system can utilize many different hardware devices. Multiple terminals, modems, printers, plotters, and disks are just some of the devices that can be connected to the system.

The minimum HP-UX system consists of the following:

- A computer (called the system processing unit or SPU).
- A system console.
- Some random-access memory (RAM).
- A mass storage disk.
- A tape drive (1/4-inch cartridge, 1/2-inch magnetic drive, or DAT) or CD-ROM.

The computer, of course, is the main piece of hardware. The UNIX operating system runs on many different types of computers, from PCs to mainframes. HP-UX requires that you have a Hewlett-Packard 9000 Series 300, 400, 700, or 800 computer.

Many UNIX operating systems require that a system console be installed. A terminal or graphics display is most frequently used as the system console. The system console is used for displaying boot up messages and for entering commands during a system installation or update. Once the UNIX system is installed and running, the operating system uses the system console to allow an administrator to access the system in single-user mode. The console is also used to display certain system error messages and can be used as an ordinary terminal on the machine.

HP-UX requires that you have a system console installed. If you are working on a graphics workstation, the graphics display is normally the system console.

The RAM in the computer is where programs get loaded to be executed. Simple programs and applications will not use as much system memory as larger, more complex programs.

The disk is used for storing all of the programs and data files that the system uses. The UNIX system is called a disk-based operating system. This means that it requires a disk in order to function. One of the operating system's biggest jobs is controlling access to the files on the disk.

Tape drives and CD-ROMs are used to load the system software when your system is new (some workstations may come with the operating system pre-installed). They are also used for making backups of your programs and data files so if something should go wrong with your system, you will not lose your work.

2-1. SLIDE: UNIX System Hardware

Instructor Notes

Purpose

A UNIX system can utilize many different hardware devices. This is what a typical system would look like.

Key Points

- Regardless of the version of HP-UX being used, the minimum system consists of a Series 300, 400, 700, or 800 computer, a system console, a mass storage disk, and a tape or flexible disk drive.
- The HP-UX system requires that a system console be installed. This is usually an HP terminal or graphics display. HP-UX does not require that the device be turned on.
- Some workstations come with the operating system already installed.

Teaching Question

What is the minimum hardware necessary to run HP-UX?

Answer: a computer, a system console, RAM, a mass storage disk, a tape drive.

Positive Instance

Although many computers will have the capability to accommodate many terminals, printers, plotters, and network communications, every HP-UX computer system will minimally have an SPU, a system console, some amount of RAM, a mass storage disk, and a secondary storage device installed.

All error messages generated by HP-UX are sent to the system console.

Negative Instance

An HP-UX system cannot run without any of its minimum components. For example, if a system console is not installed, there is no way an administrator can interact with the system nor can HP-UX output any error messages. HP-UX cannot tolerate this situation.

Where Problems Arise

It may be necessary to make a distinction between HP-UX on the Series 300, Series 400, Series 700, and Series 800 and the UNIX operating system on other types of computers. The minimum hardware necessary is slightly different for each machine and the different versions of the UNIX operating system.

Slide Notes

The dashed lines indicate that these peripherals are optional.

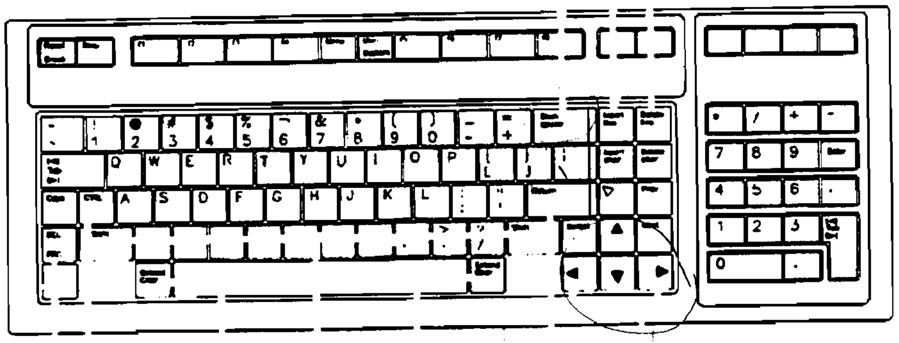
Transition

An overview of terminals is presented next.

2-2. SLIDE: Terminals

Terminals

- Both alphabetic and graphics terminals.
- Terminals are regarded as peripheral devices.
- Number of active terminals is based upon system's license.



The HP ITF Keyboard

51489 2-2 6 © 1991 Hewlett-Packard Company

backspace [illegible] cursor keys

Student Notes

All terminals are treated as peripheral devices. The maximum number of terminals permissible on the system is limited by the system's user license. This license determines how many people can access the system concurrently. Common licenses are 2-user, 8-user, 16-user, 32-user, 64-user, 128-user, 256-user, and unlimited-user.

Most of the keys on the keyboard are like those on a standard typewriter keyboard. You can type uppercase letters, lowercase letters, and symbols.

Unlike a typewriter, there is a difference between a lower case "l" and a "1" (one), and between the letter "O" and a "0" (zero).

The UNIX operating system is case sensitive for all commands and file names.

A given application may redefine some other keys on the keyboard, so be sure to check your application documentation for details.

2-2. SLIDE: Terminals

Instructor Notes

Purpose

Users should be familiar with their supported terminals.

Key Points

The UNIX operating system treats all terminals as peripheral devices. The maximum number of active terminals permissible on the system is limited by the system's user license.

Teaching Question

Is there a limitation as to the type of terminal that can be used with an HP-UX System?

Positive Instance

Both alphabetic displays and graphics terminals can be used.

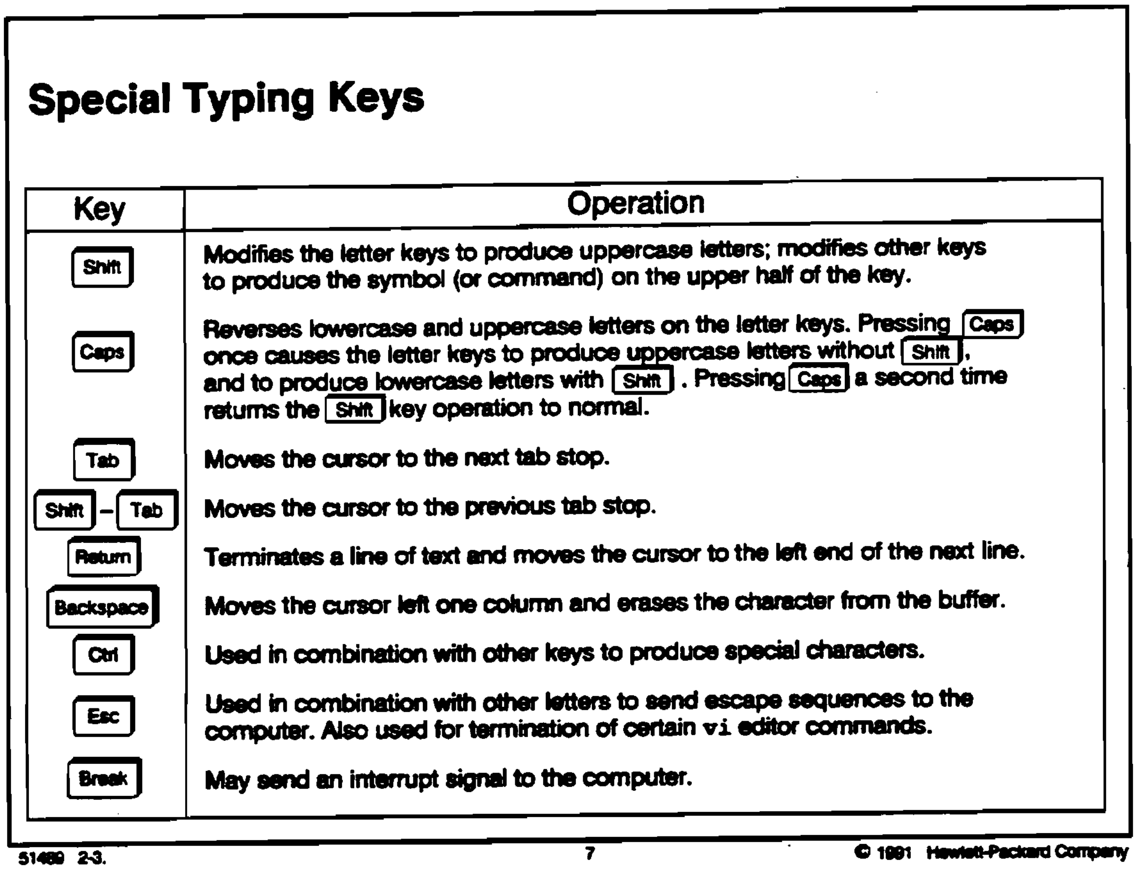
Where Problems Arise

As with other hardware devices, prolonged discussion of individual model numbers is not necessary. The student should appreciate the variety of terminals available for the Series 300, 400, 700, and 800 and can find detailed information on each in the *HP 9000 Configuration Guide*.

Transition

Users should be familiar with the terminal's special typing keys.

2-3. SLIDE: Special Typing Keys



Special Typing Keys

Key	Operation
Shift	Modifies the letter keys to produce uppercase letters; modifies other keys to produce the symbol (or command) on the upper half of the key.
Caps	Reverses lowercase and uppercase letters on the letter keys. Pressing Caps once causes the letter keys to produce uppercase letters without Shift, and to produce lowercase letters with Shift. Pressing Caps a second time returns the Shift key operation to normal.
Tab	Moves the cursor to the next tab stop.
Shift - Tab	Moves the cursor to the previous tab stop.
Return	Terminates a line of text and moves the cursor to the left end of the next line.
Backspace	Moves the cursor left one column and erases the character from the buffer.
Ctrl	Used in combination with other keys to produce special characters.
Esc	Used in combination with other letters to send escape sequences to the computer. Also used for termination of certain `vi` editor commands.
Break	May send an interrupt signal to the computer.

51489 2-3. 7 © 1991 Hewlett-Packard Company

Student Notes

These keys are used to manipulate text and command lines. You should be familiar with these keys and their functions in different applications. You will be using these keys throughout the course.

2-3. SLIDE: Special Typing Keys

Instructor Notes

Purpose

Users should be familiar with the terminal's special typing keys.

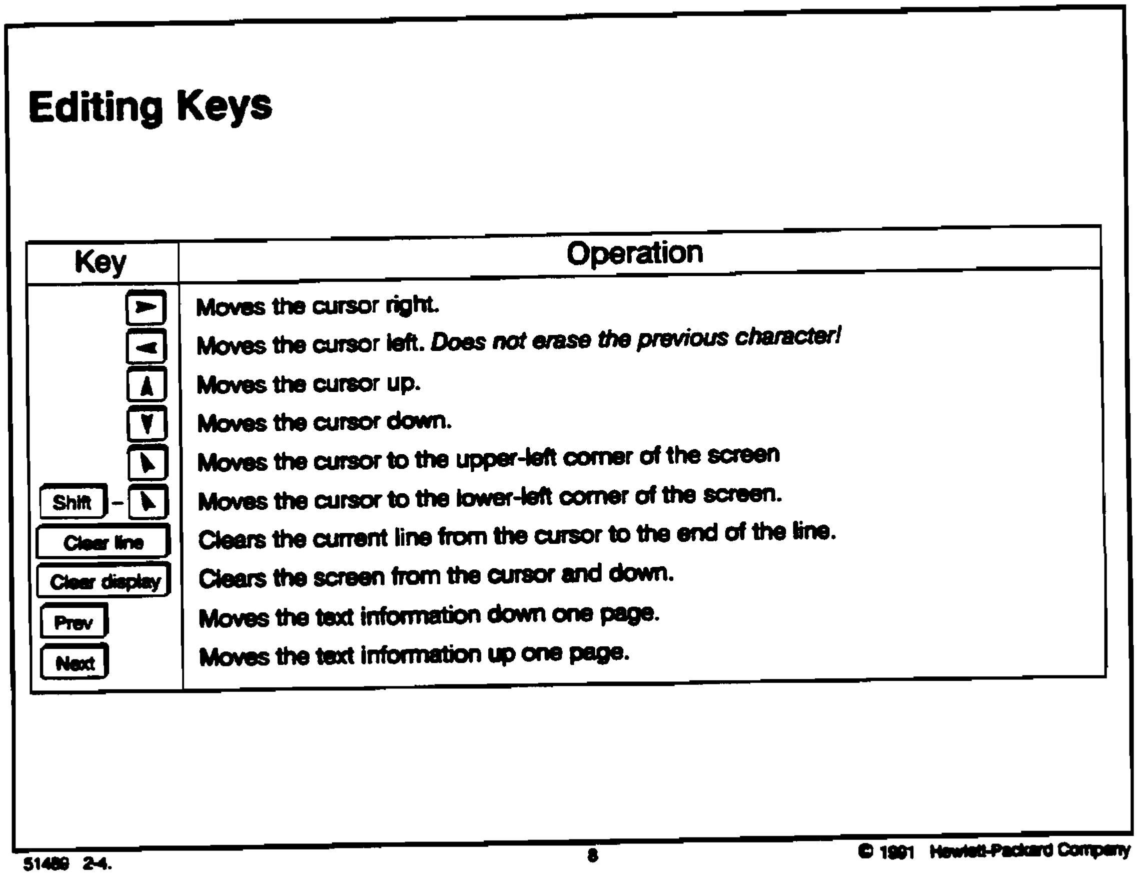
Key Points

These keys are used to modify the character keys or move the cursor. Break may not act as anticipated if `intr` is set differently. Check the `stty -a` command to see how `intr` is set.

Transition

Users should also be familiar with the terminal's special editing keys.

2-4. SLIDE: Editing Keys



Editing Keys

Key	Operation
►	Moves the cursor right.
◄	Moves the cursor left. *Does not erase the previous character!*
▲	Moves the cursor up.
▼	Moves the cursor down.
↖	Moves the cursor to the upper-left corner of the screen
Shift - ↖	Moves the cursor to the lower-left corner of the screen.
Clear line	Clears the current line from the cursor to the end of the line.
Clear display	Clears the screen from the cursor and down.
Prev	Moves the text information down one page.
Next	Moves the text information up one page.

51489 2-4. 8 © 1991 Hewlett-Packard Company

Student Notes

Using the terminal's editing keys will affect the text on your terminal screen but will not affect the operating system. For instance, the left arrow key *does not* erase characters on the command line. If you use the arrow keys to move over characters and correct typos, the computer will see both the old characters and the new ones. This will more than likely produce an error message.

Different applications will treat these keys differently. You have to be careful where and how you use them.

2-4. SLIDE: Editing Keys

Purpose

Users should be familiar with the terminal's editing keys.

Key Points

Don't spend a lot of time here! The students only need to know that these keys exist and where they should and should not use them. HP-UX does not normally recognize these keys. Applications may or may not recognize the keys. It may be dull, but it is very important to know.

These keys are used for terminal editing functions during normal computer use and are not used the same in different applications. When you enter a command or type text, you *cannot* use these keys to modify any typing mistakes unless the application you are running allows you to do so. Note that the keys will be different in applications such as the `vi` editor.

Transition

Let's identify the different components of the computer we are using here; exercises are next.

2-5. LAB: Exercises

This is an exercise in HP-UX hardware identification. For the system being used in this class, identify the following items:

1. The computer model that is being used.

2. The device that is being used as a system console.

3. The disk or disks that are connected to the system.

4. The tape device that is being used.

5. The maximum number of terminals that can be connected to the system.

2-5. LAB: Exercises

Key Points

The information the students are asked to derive is strictly dependent upon the hardware of the local system. Prior to completing this module, the instructor should prepare a listing of the hardware and software installed on the local system.

Transition

Users should learn how to log in to the computer next.

Solutions

This is an exercise in HP-UX hardware identification. For the system being used in this class, identify the following items:

1. The computer model that is being used.

Answer:

The computer model used is:

2. The device that is being used as a system console.

Answer:

The device is:

3. The disk or disks that are connected to the system.

Answer:

The disk or disks connected to the system are:

4. The tape device that is being used.

Answer:

The tape device being used is:

5. The maximum number of terminals that can be connected to the system.

Answer:

The maximum number of terminals is:

Module 3 — Logging in and Orientation

Objectives

Upon completion of this module, you will be able to do the following:

- Log in to and log off of a UNIX system.
- Explain the need for and be able to use the erase and line kill characters.
- Execute commands with and without arguments.
- Change your password.
- Use the `date` and `who` commands to obtain system information.
- Use the `mailx` and `news` commands for communicating with system users.
- Use the `echo` and `banner` commands for displaying text.
- Use the *HP-UX Reference Manual* to explain the purpose of each section.
- Access the online copy of the *HP-UX Reference Manual*.

[illegible]

UNIX [illegible]

[illegible]

[illegible]
- [illegible]
- [illegible]

- [illegible]
  - [illegible]
  - [illegible] information - 'quiet'

- [illegible] standard [illegible]

[illegible]

[illegible]
THREE [illegible]

[illegible]

- [illegible] arrange a series of [illegible]
  > prepared command → [illegible] script
  (simple executable text file)

- [illegible]

- [illegible]

- [illegible]

Module 3 — Logging in and Orientation

Overview of Module 3

This module provides the students with their first taste of terminal activity with the HP-UX shell.

This module introduces the general log in/log off sequence, the erase and kill characters, and some simple commands (`date`, `who`, `mailx`, `news`, `echo`, `banner`, and `man`).

It should be pointed out that the UNIX system normally works with ASCII characters and that all shell commands use lowercase letters.

Motivation

Students should start by learning the UNIX system basics. The more important shell criteria are introduced.

Exercises

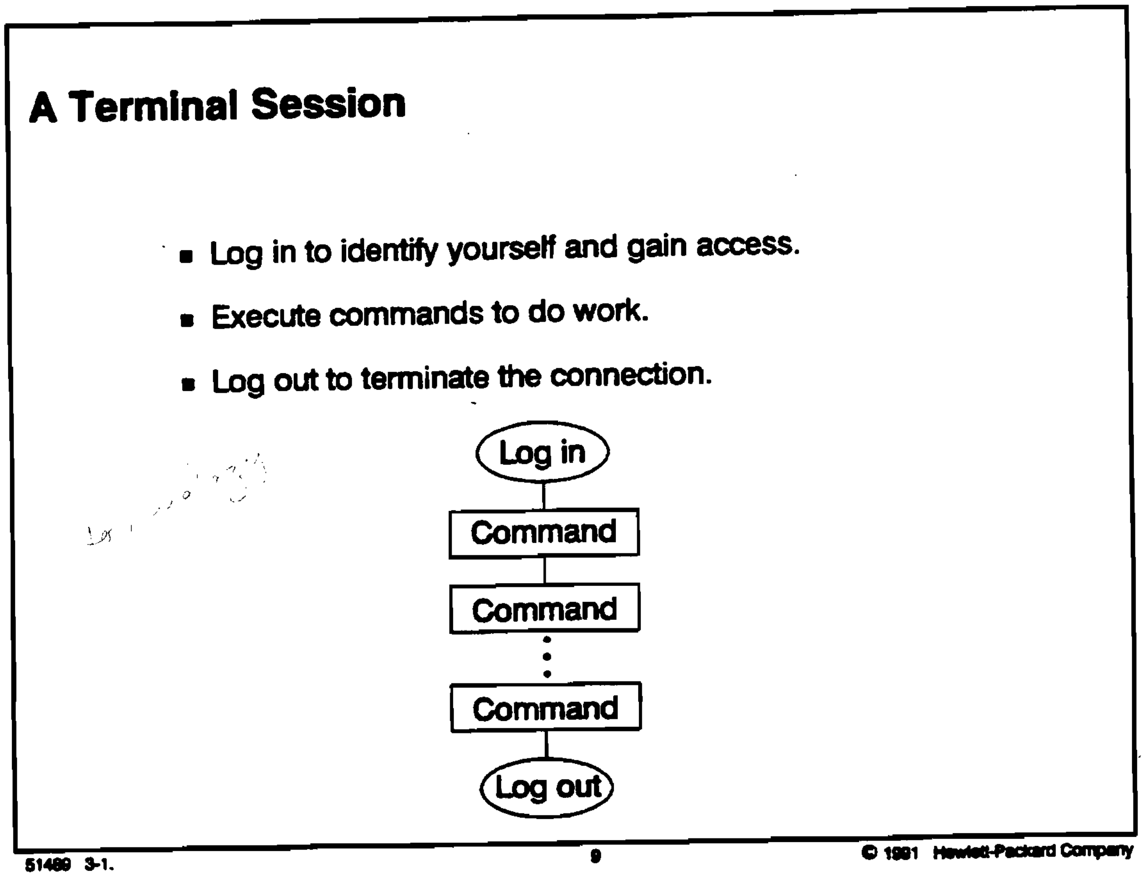
The terminal exercises are dispersed throughout this module. Each student should be assigned a login id by the time they are asked to do the exercises.

If you have a large class or a class that requires a lot of special attention, you may want to save all exercises until the end of the module. Interspersing the exercises works well in small or very sharp groups. Otherwise it may slow you down.

Module Transition

The next step in becoming a knowledgeable user is to learn something about the file system. This is how the UNIX operating system organizes the disk so we can keep track of our files.

3-1. SLIDE: A Terminal Session



A Terminal Session

- Log in to identify yourself and gain access.
- Execute commands to do work.
- Log out to terminate the connection.

Log in
Command
Command
Command
Log out

51489 3-1. 9 © 1991 Hewlett-Packard Company

Student Notes

A terminal session begins by logging in and ends by logging off. In between, you can execute commands to do work on the computer. A command is the only way to make the computer do something.

The way the UNIX system identifies the many users on the system is by their **user name** (sometimes called the **login id**). When you sit down at a terminal, there will be some sort of login prompt on the screen. This is where you type your user name. If you have a password set, it will also ask you for the password before letting you on to the system.

Once you are logged in, you can enter commands and the operating system will execute them on your behalf.

Logging off the system is the process of closing your terminal connection. When you log off the system, a new login prompt will appear so another user can log in.

3-1. SLIDE: A Terminal Session

Instructor Notes

Purpose

To prime the students for what a terminal session is (log on, do work, and log off).

Key Points

Users cannot access the system's programs or files until they log in.

Teaching Question

Ask students to explain when a terminal session begins.

Answer: A terminal session begins by logging in.

Positive Instance

Explaining the login process is simply making the connection (recognized by the `login:` prompt) and entering proper login identification (id) and password.

The user will gain access to the system if the correct user name and password are given.

Negative Instance

Users cannot log in if the login and password do not match what the system expects for them.

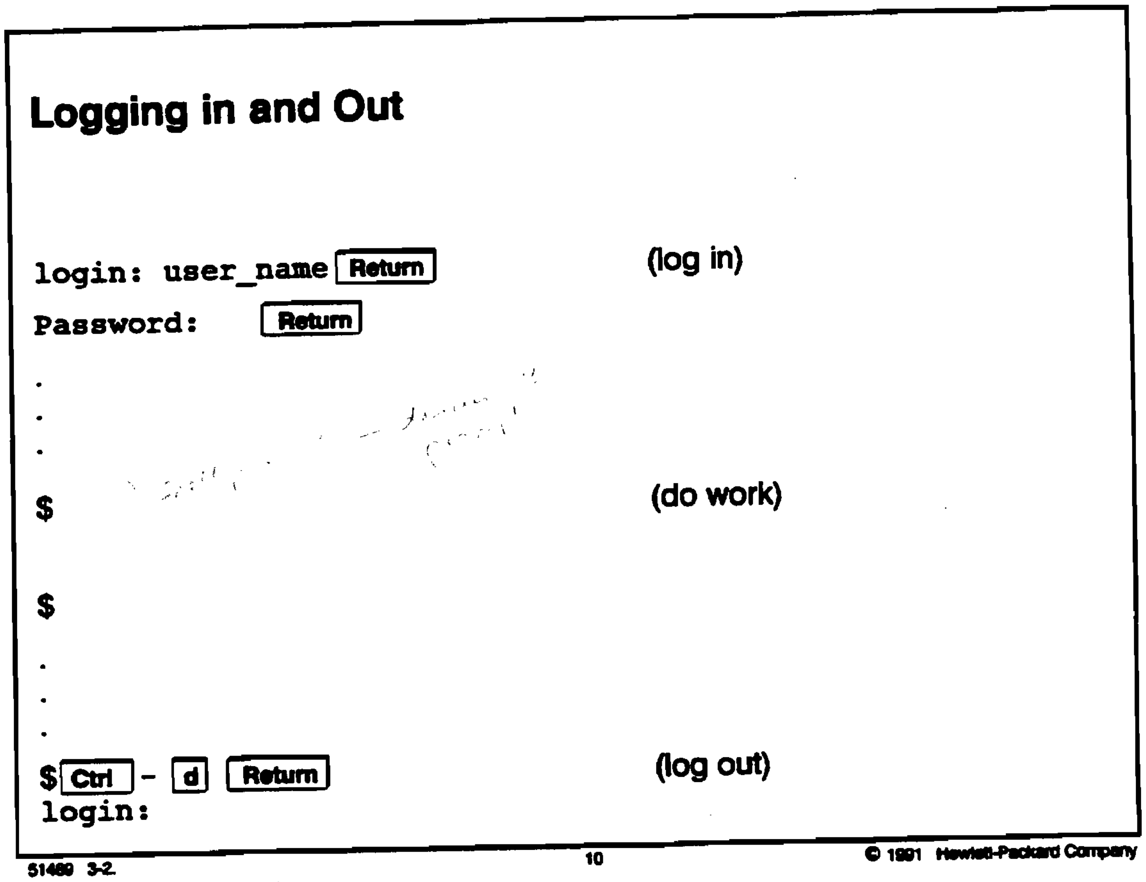
Evaluation Questions

Ask if anyone has questions about establishing a terminal session.

Transition

Now that you have the general understanding of what is required to access the system, let's look in detail at how to log in and out.

3-2. SLIDE: Logging in and Out



Logging in and Out

```
login: user_name [Return]          (log in)
Password:    [Return]
.
.
.
$                                  (do work)

$
.
.
.
$[Ctrl] - [d] [Return]             (log out)
login:
```

51480 3-2 10 © 1991 Hewlett-Packard Company

Student Notes

Your login id is assigned to you by a system administrator and is normally your name, initials, or some other unique identifier. You may also have a password assigned. This password is yours alone. You determine your password, and absolutely no one can find out what it is. If you forget your password, you have to go back to the system administrator and ask him or her to change it for you.

To log in, do the following:

1. Turn on the terminal. Some terminals have display timeouts so you may have to press a key to reactivate the display.
2. If you do not get the `login:` prompt or you get "garbage" printed, press (Break). The "garbage" usually means that the computer is trying to communicate with your terminal at the wrong speed. (Break) tells the computer to try another speed. You can press (Break) repeatedly, thus trying different speeds, but wait for a response each time.
3. When the `login:` prompt appears, type your login id and press (Return).

4. If you have a password set, the `Password:` prompt appears. Type your password and press Return. Notice that your password is not displayed on the screen while you type it.

If you did not enter the correct login id and password combination, the system will tell you that the login attempt failed and will issue another login prompt. If you did provide the correct combination, the "shell prompt" will be printed on your screen. This indicates that the computer is ready to accept your commands so you can do work.

When you are finished using the system, you need to log off to close your connection to the system. To do this, type Ctrl-d Return or `exit` followed by Return at the shell prompt. The notation Ctrl-d indicates that you type a "control d." This is done by holding down the Ctrl key while pressing d (lower case letter "d"). Note that the Ctrl-d must be typed at the beginning of the line after the shell prompt.

You will know that you are logged off when a login prompt appears on your screen.

3-2. SLIDE: Logging in and Out

Instructor Notes

Purpose

The student will see a login flow. The user will know if the login attempt was successful when the shell prompt is displayed at the terminal.

Key Points

Explain the proper method of logging off is to enter Ctrl-d followed by Return as the first (and only) character after the shell prompt. You can also log off by typing `exit` and pressing Return.

The `ksh` requires that you type Ctrl-d Return to log out.

Where Problems Arise

Students are usually confused when they use the Backspace key when typing their login id or password and the system tells them the login is incorrect. Show the next slide before letting the students log in.

Chalkboard Notes

Provide the students with login ids (user names) and passwords. If you are concerned about student control, delay giving the login and password information until later in this module (for example, at the beginning of the terminal exercises).

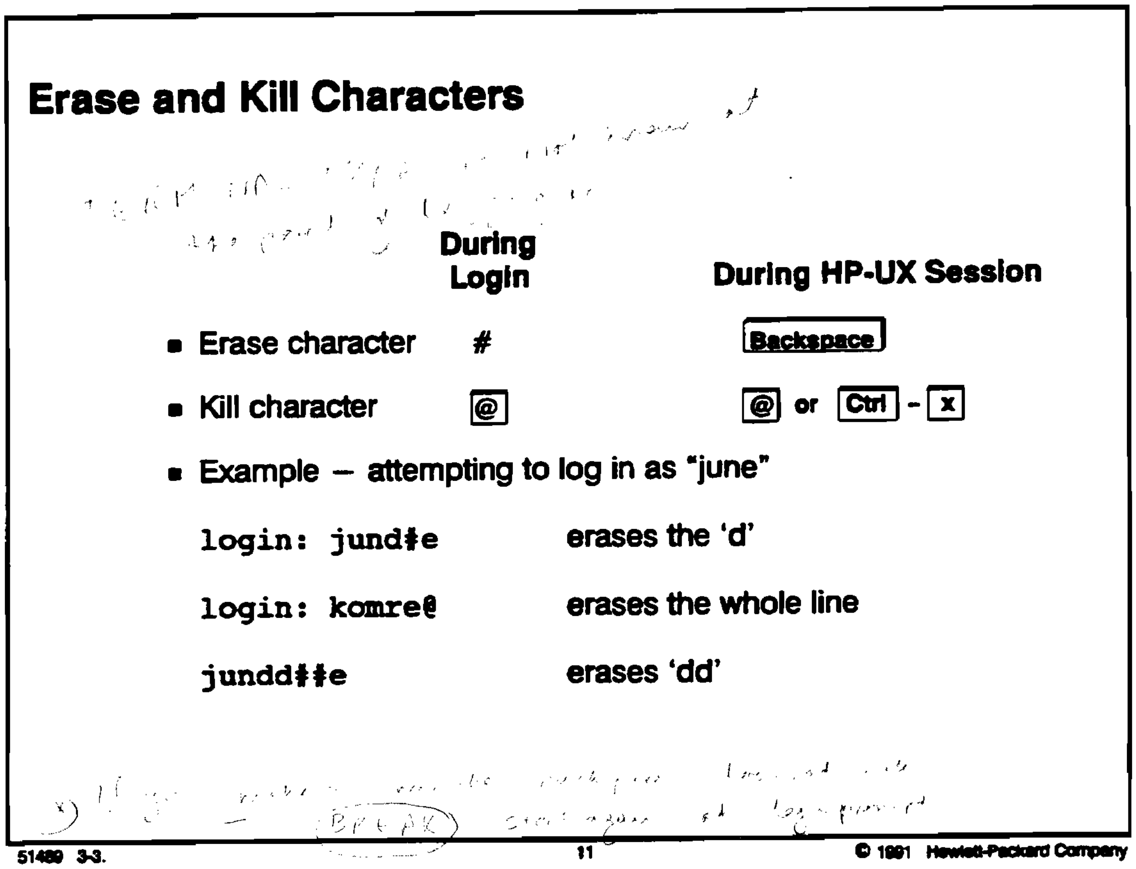
Evaluation Questions

Ask the students to explain when they will know they can enter a command line and have it executed. Also ask for the correct method of logging off the system.

Transition

In the next slide, erasing characters is discussed.

3-3. SLIDE: Erase and Kill Characters



Erase and Kill Characters

[illegible]

	During Login	During HP-UX Session
■ Erase character	#	Backspace
■ Kill character	@	@ or Ctrl - x

■ Example – attempting to log in as "june"

`login: jund#e`	erases the 'd'
`login: komre@`	erases the whole line
`jundd##e`	erases 'dd'

[illegible] BREAK [illegible]

51489 3-3. 11 © 1991 Hewlett-Packard Company

Student Notes

For both login and password, the # acts as a backspace, and the @ backspaces over the entire line. Be careful, because the backspace key does not work while you are logging in.

These special "erase and kill characters" are used when you log in because not all terminals use the same method of backspacing. By using # and @ as the erase and kill characters, you can make corrections during login regardless of what kind of terminal you are using.

Remember, while you are logging in:

- # deletes the previous character.
- @ deletes the current line (just like backspacing over the whole line).

Once you have logged in, the system identifies your terminal and the proper erase and kill characters are set. Then, for the rest of your session:

- (Backspace) is the erase character, which deletes the previous character.
- (@) or (Ctrl)-(x) is the kill character for users, which erases the current line and allows you to start over.

Your installation may have different characters set for the erase and kill characters. Check with your system administrator to find out what these special characters are.

You do not need to use erase and kill when you are logging in. If you do not type your login id correctly and then press (Return), the system will still ask you for a password. Just press (Return) and the system will tell you that the login is incorrect and give you another login prompt.

If you log in using uppercase letters, HP-UX will think you are on an all uppercase terminal and lock your terminal in uppercase mode. This means that you cannot use any command or file that contains an uppercase letter in its name.

To reset this lock, you can type:

```
$ STTY -LCASE
$
```

This will restore you to normal uppercase and lowercase mode.

Exercise

Try to log in.

Please, do not do anything else yet.

3-3. SLIDE: Erase and Kill Characters

Instructor Notes

Purpose

Terminal interaction is one character-at-a-time, so we need to have the capability of erasing characters and complete command lines correctly.

Key Points

The erase character erases the previous character, the kill character erases the current line. Different characters are assigned to these functions for logging in versus the rest of the terminal session.

Make sure you tell students what the erase and kill characters are on the classroom systems.

Teaching Question

Based upon students' current understanding of the command line, ask them what the shell will interpret as the command for the line "oranges@apples . . . ".

Answer: apples

Where Problems Arise

Students sometimes have trouble understanding why we need different erase and kill characters when we log in. Stress that the UNIX operating system supports many different kinds of terminals and many use different keys for erase and backspace. (In fact, some terminals supported by the UNIX operating system do not even have backspace keys!)

Emphasize the fact that the kill character does not cause a new shell prompt to be issued. It is confusing initially when they see the cursor return to the beginning of the "next" line rather then give them a new prompt.

Also emphasize the fact that the erase and (especially) the kill characters may be set differently on their systems. You may need to point out that the `stty -a` command will allow them to see how the keys are set.

Evaluation Questions

Determine whether the students have any questions about when they might want to use either of these special shell characters.

Presentation Suggestions

Emphasize the difference in the erase and kill characters between logging in and the actual terminal session.

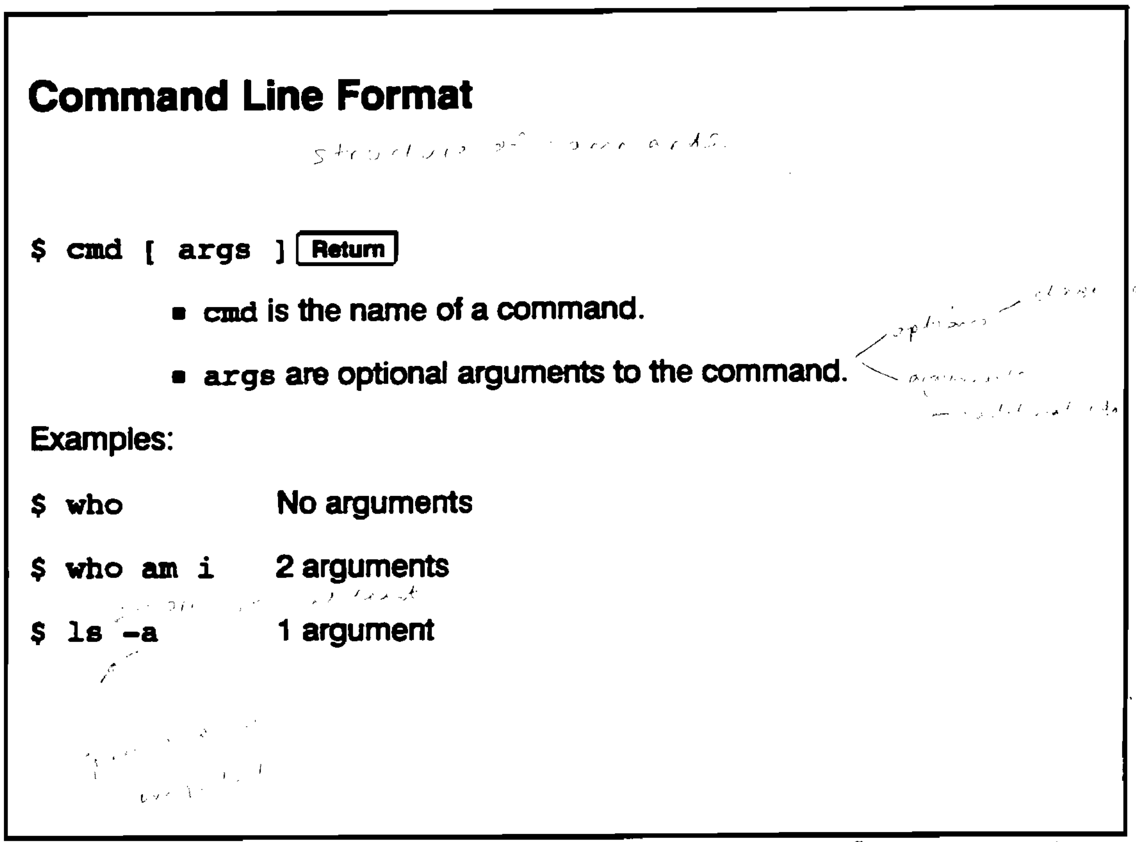
Chalkboard Notes

A few chalkboard examples usually help if students are having trouble.

Transition

Now that a basic understanding of typing in the UNIX system has been established, it's time to take a look at the command line.

3-4. SLIDE: Command Line Format



Command Line Format

structure of commands

```
$ cmd [ args ] Return
```

- cmd is the name of a command.
- args are optional arguments to the command. [illegible]

Examples:

```
$ who          No arguments
$ who am i     2 arguments
$ ls -a        1 argument
```

51489 3-4. 12 © 1991 Hewlett-Packard Company

Student Notes

A command is a computer program that performs a specific task. You execute the commands you need to do your work on the system.

A command line consists of a group of words that you type at your shell prompt. The command name is the *first* word on the line. Other words on the command line are **arguments** to the command. Arguments are a way of telling a command what you want to do. Some commands have no arguments while others have several.

White space is used to delimit (separate) the commands and arguments on the command line. White space means leaving one or more blank spaces or tabs between the words on the command line.

To transmit the command to the computer, you must end the command line with a carriage return; in other words, you must press Return. A minimal command line would be the command name typed at the shell prompt followed by Return.

Commands in the UNIX operating system have short names, for example: `banner`, `who`, `cat`, `ls`. (We will examine these commands and many others.) If you are not a great typist or you work at the terminal for long stretches, you will appreciate the terseness of these commands.

Note The UNIX operating system is the embodiment of a philosophy of the way computer systems ought to work. The philosophy states that there should be a wide variety of commands, each of which does one thing well, and the user should be free to combine the commands to perform a variety of tasks. Commands should not second guess the user and should not clutter the terminal with verbose error messages or acknowledgments. Thus, with few exceptions, commands will not attempt to protect you from yourself.

HP-UX also follows this philosophy. Normally, if you do not receive an error message on your screen, the command worked correctly.

3-4. SLIDE: Command Line Format

Instructor Notes

Purpose

Because the shell interprets items entered after the prompt in a specific sequence, it is important to understand what comprises a command line.

Key Points

At least two "items" are required for the shell to perform a meaningful action. The name of a valid command must be typed followed by a carriage return.

Teaching Question

Ask the students to explain how the shell can distinguish a command from its arguments.

Answer: A white space separates a command from its arguments.

Positive Instance

The shell is able to execute the command line.

Negative Instance

The shell is not able to execute the command line because something was entered incorrectly (perhaps a typographical error in the command name).

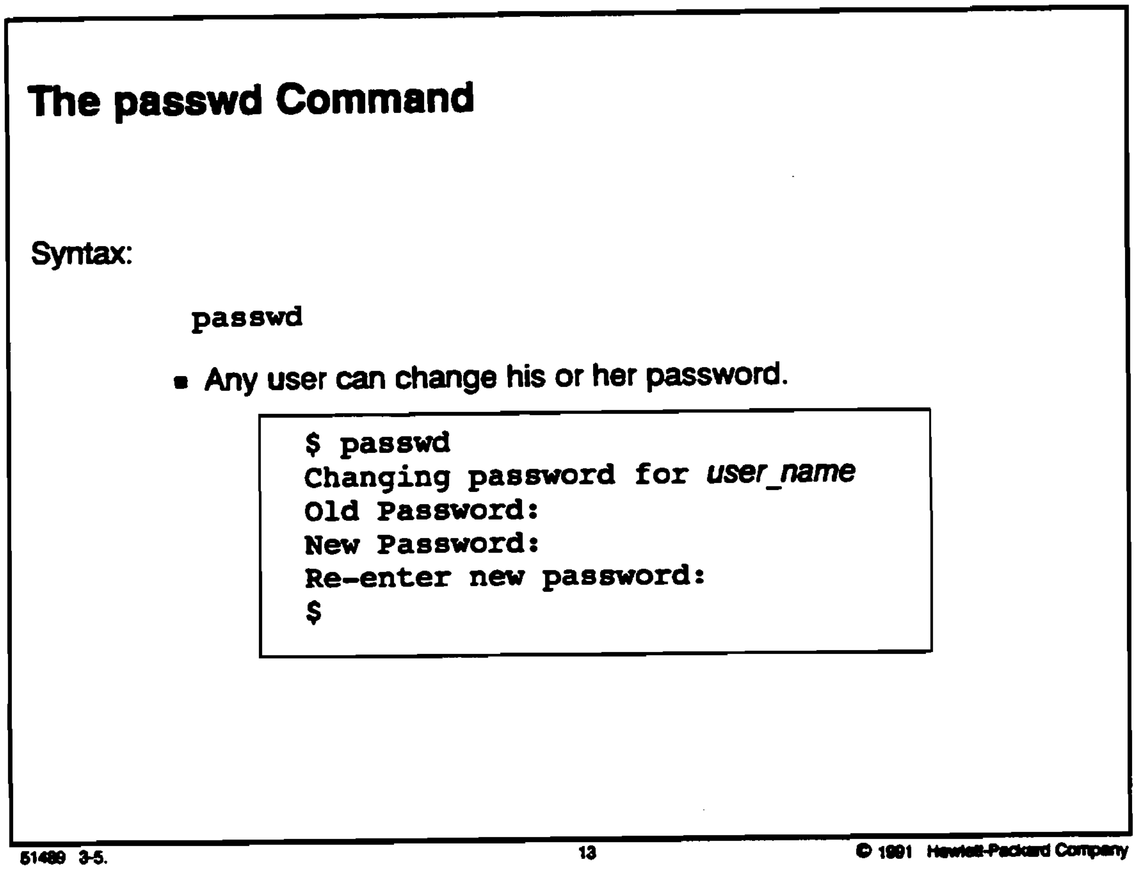
Where Problems Arise

Beginners have a tendency to forget to enter white space or use the wrong command and hence do not understand why their command did not execute correctly.

Transition

Let's take a look at some common HP-UX commands.

3-5. SLIDE: The passwd Command



The passwd Command

Syntax:

```
passwd
```

- Any user can change his or her password.

```
$ passwd
Changing password for user_name
Old Password:
New Password:
Re-enter new password:
$
```

51489 3-5. 13 © 1991 Hewlett-Packard Company

[illegible]

Student Notes

To establish or change your password, use the passwd command. After invoking this command, you will be asked for your old password (if you had one), and then you will be asked to type your new password twice. The second time is to verify that you did not misspell your password. Note that none of the characters you type will show on the screen.

No one on the system can find out your password. If you forget it, the system administrator will have to set a new one for you.

All passwords must be at least six characters of your choosing. The password must contain at least two alphabetic characters and one numeric or special character within the first eight characters. The system administrator is the only user who does not need to follow these rules.

[illegible]

3-5. SLIDE: The passwd Command

Purpose

For better security, we need to be able to change our passwords when desired.

Key Points

All passwords must be at least six characters of your choosing, containing at least two alphabetic characters and one numeric or special character.

Teaching Question

Why does the system prompt you for your new password twice?

Answer: To verify that you did not misspell your password.

Where Problems Arise

The restrictions on password length and content can be confusing. The `passwd` command will tell you if the password is invalid and why.

Slide Notes

Point out that the "`$`"s on the slide are the shell prompts and should not be typed.

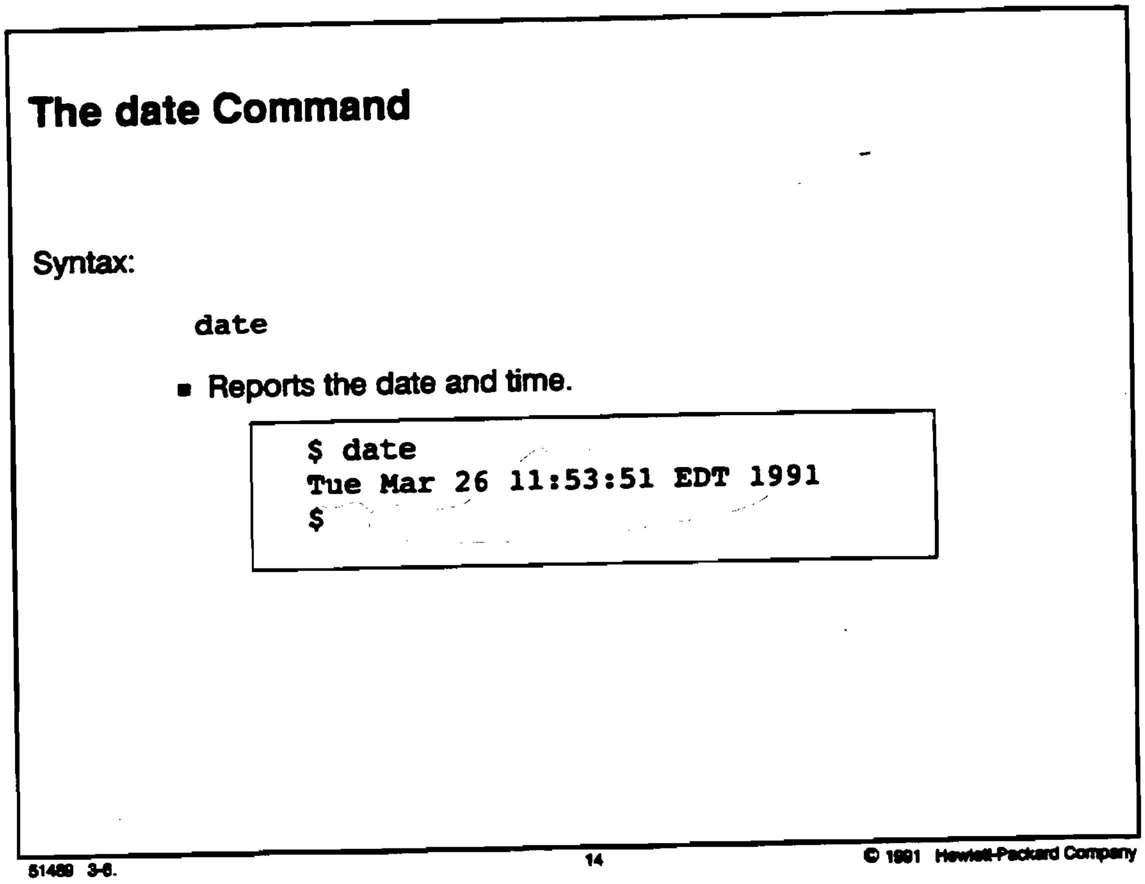
Chalkboard Notes

Write some examples of passwords on the board. Use both valid and invalid examples and have students determine which ones are invalid and why.

Transition

A command that is used more often is the `date` command.

3-6. SLIDE: The date Command



The date Command

Syntax:

```
date
```

- Reports the date and time.

```
$ date
Tue Mar 26 11:53:51 EDT 1991
$
```

51489 3-6. 14 © 1991 Hewlett-Packard Company

Student Notes

Here is an example of a simple command. The `date` command is used to report the date and time. It may have arguments that affect the format of the output. There is also an argument to set the date and time, but only the system administrator can use this argument.

Since the `date` command is most often used with no options or arguments, that is how we present it here. Remember that the "`$`" is the shell prompt where we enter our commands.

Exercise

1. You should still be logged in. Try to report the date and time by executing the `date` command at the shell prompt.

3-6. SLIDE: The date Command

Instructor Notes

Purpose

There are basically three uses of the date command: `date` without an argument, `date` with one argument to change the system's date or time or both (system administrator only), and `date` with one argument any user can enter to format the date command output.

Key Points

The `date` command is used interactively by the user to get the current time. When it is used in a shell program, it will most likely be formatted.

Evaluation Questions

Ask whether any students have any specific questions or problems with the command.

Advanced Topic

You may also mention the `cal` command for displaying calendars. `leave` is another fun command that deals with the date.

Transition

Another frequently used command tells the user who else is currently logged in.

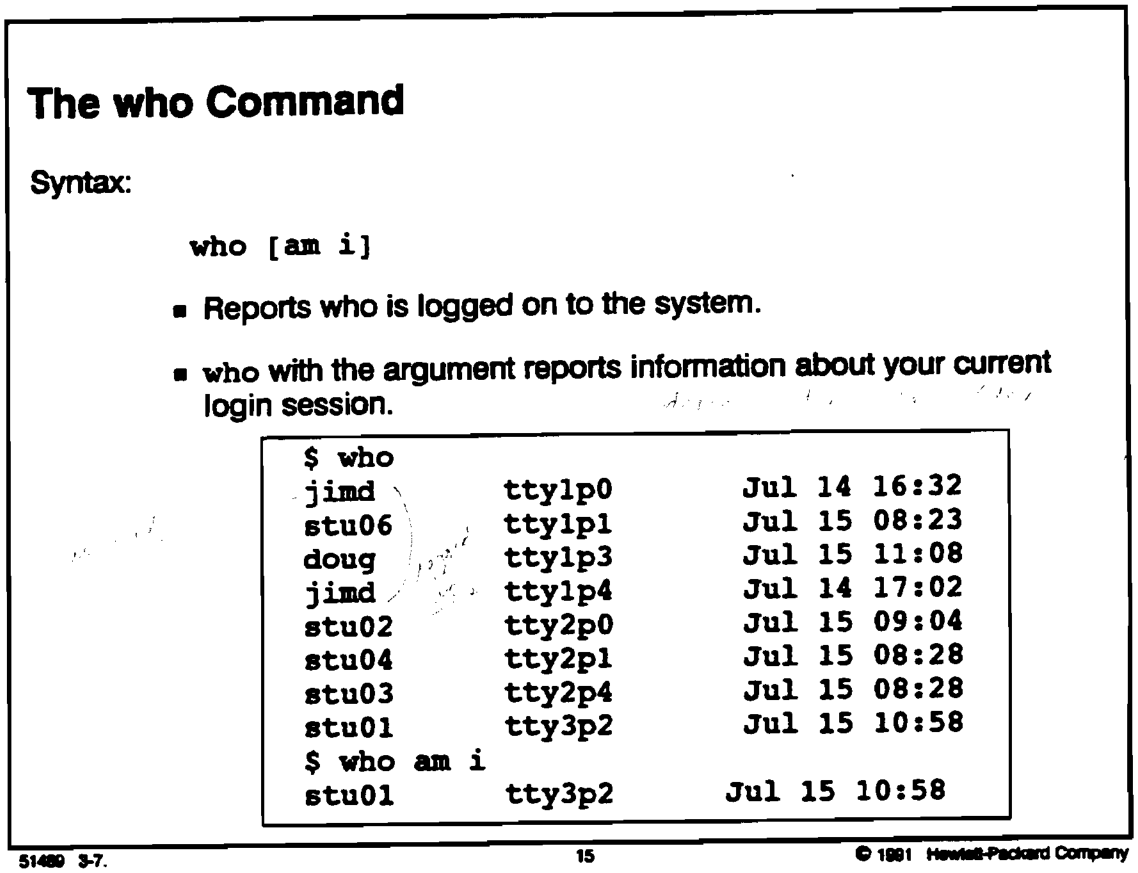
Answer

1. You should still be logged in. Try to report the date and time by executing the `date` command at the shell prompt.

Answer:

```
$ date
```

3-7. SLIDE: The who Command



The who Command

Syntax:

```
who [am i]
```

- Reports who is logged on to the system.
- who with the argument reports information about your current login session.

```
$ who
jimd        tty1p0      Jul 14 16:32
stu06       tty1p1      Jul 15 08:23
doug        tty1p3      Jul 15 11:08
jimd        tty1p4      Jul 14 17:02
stu02       tty2p0      Jul 15 09:04
stu04       tty2p1      Jul 15 08:28
stu03       tty2p4      Jul 15 08:28
stu01       tty3p2      Jul 15 10:58
$ who am i
stu01       tty3p2     Jul 15 10:58
```

51480 3-7. 15 © 1991 Hewlett-Packard Company

Student Notes

The who command reports which users are logged in to the system, what terminal they are using, and when they logged in. This is the most common use of the who command:

```
$ who
june        tty1p4      Jul 26 06:43
root        tty1p5      Jul 27 07:42
gerry       tty3p2      Jul 25 16:07
jimd        tty3p3      Jul 27 17:29
```

If you give the arguments am and i to the who command, it just displays your line from the who output.

```
$ who am i
gerry       tty3p2      Jul 25 16:07
```

About the slide: Remember, the dollar signs are shell prompts. Notice that it is possible for a user to be logged in at more than one place.

Exercise

1. Try each of the following commands and note the difference in output: `who`, `who am i`.

3-7. SLIDE: The who Command

Instructor Notes

Purpose

We sometimes need a method of determining who else is logged in to the system.

Key Points

Note the difference between `who` (no arguments) and `who am i` (`who` with two arguments).

Evaluation Question

Ask how many arguments each of the exercise commands has.

Where Problems Arise

Notice that in the slide `jimd` appears twice. Users may be logged in at more than one terminal. When using windows, each window running a shell will be reported as another line in the `who` output.

Slide Notes

Use "reveal" (covering lines of text with a sheet of paper and then moving the sheet lower) to display the command line and output one at a time.

Transition

The `echo` command is a command that writes its arguments to the terminal.

Answer

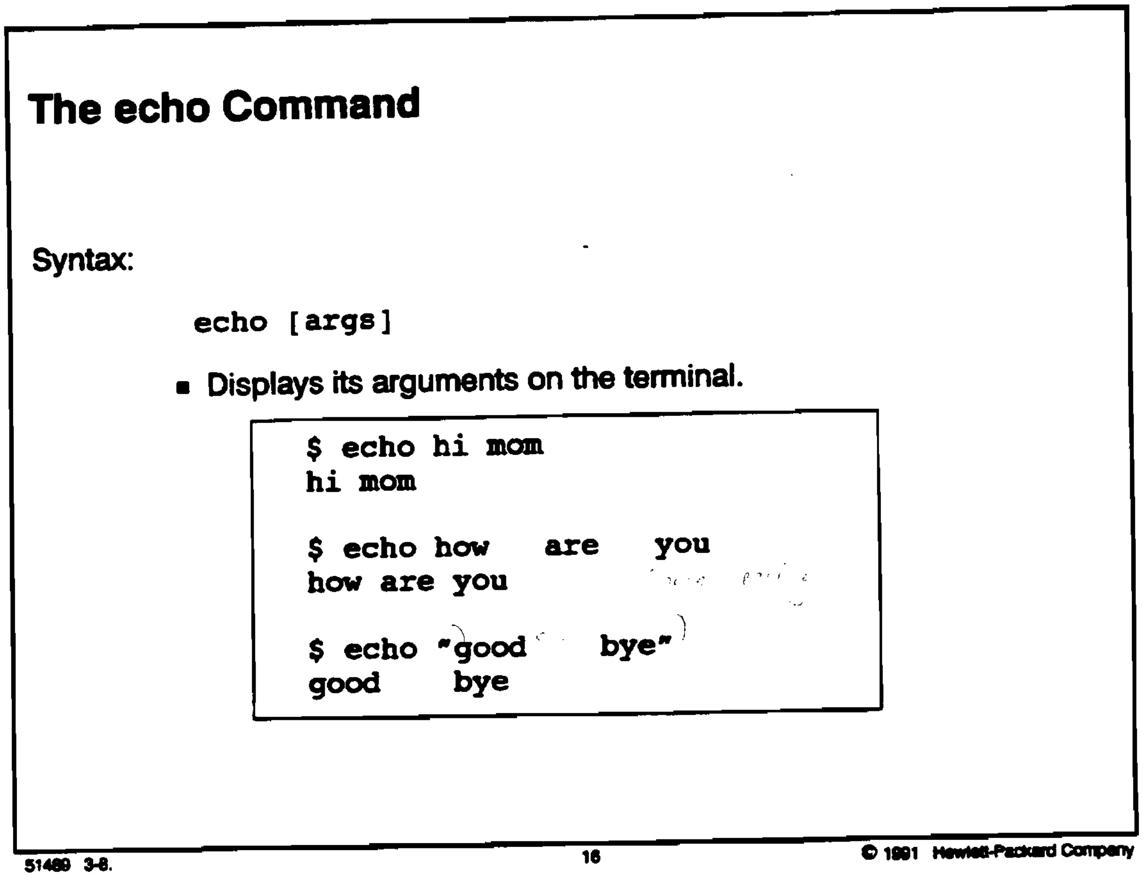
1. Try each of the following commands and note the difference in output: `who`, `who am i`.

Answer:

You should see output similar to the examples on the slide:

```
$ who
gerry      console      May 25 10:06
kevin      tty1p0       May 25 13:10
ginger     tty1p1       May 25 08:04
renate     tty1p2       May 24 08:29
kevin      pty/ttyp2    May 25 10:10
$ who am i
kevin      pty/ttyp2    May 25 10:10
```

3-8. SLIDE: The echo Command



The echo Command

Syntax:

```
echo [args]
```

- Displays its arguments on the terminal.

```
$ echo hi mom
hi mom

$ echo how   are   you
how are you

$ echo "good    bye"
good    bye
```

51489 3-8. 16 © 1991 Hewlett-Packard Company

Student Notes

The echo command writes its arguments to the display and follows them with a newline. If there are no arguments, it just writes a newline. At this point it might seem a somewhat useless command, but it has its applications in shell programming.

The echo command also recognizes a few special arguments which control the action of the cursor. For example, you can tell it not to issue the newline by giving it the argument \c.

3-8. SLIDE: The echo Command

Instructor Notes

Purpose

To introduce the echo command.

Key Points

The number of white spaces between the arguments does not affect the output of the command. The double quotation marks around good and bye makes the string appear as one argument because the white space between the words is interpreted as part of the string instead of as a delimiter.

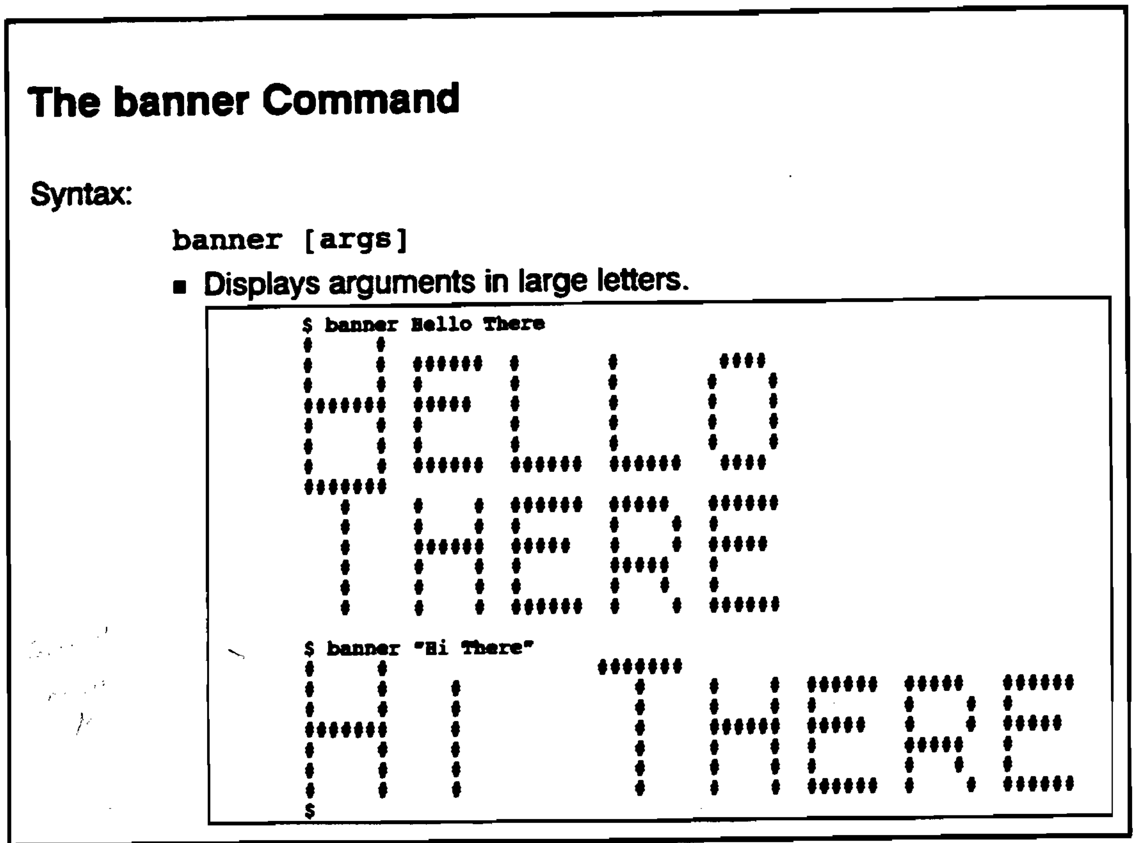
Chalkboard Notes

Give some examples of echo commands, arguments, and output.

Transition

The banner command is a fun command that prints big letters.

3-9. SLIDE: The banner Command



The banner Command

Syntax:

```
banner [args]
```

- Displays arguments in large letters.

```
$ banner Hello There
#     #
#     #  ######   #        #        ####
#     #  #        #        #       #    #
#######  #####    #        #       #    #
#     #  #        #        #       #    #
#     #  #        #        #       #    #
#     #  ######   ######   ######   ####
#######
   #     #     #  ######   #####   ######
   #     #     #  #        #    #  #
   #     #######  #####    #    #  #####
   #     #     #  #        #####   #
   #     #     #  #        #   #   #
   #     #     #  ######   #    #  ######

$ banner "Hi There"
#     #                  #######
#     #      #              #     #     #  ######   #####   ######
#     #      #              #     #     #  #        #    #  #
#######      #              #     #######  #####    #    #  #####
#     #      #              #     #     #  #        #####   #
#     #      #              #     #     #  #        #   #   #
#     #      #              #     #     #  ######   #    #  ######
$
```

51489 3-9. 17 © 1991 Hewlett-Packard Company

Student Notes

The **banner** command displays its arguments on your screen in large letters. Notice that each argument is printed on a separate line. If an argument is longer than 10 characters, only the first 10 characters will be displayed.

If you want to display more than one word on a line, you have to make the words look like one argument. An example of the **banner** command is given on the slide.

In the first example, there are two arguments and thus two lines of output. In the second example, there are quotes around the words to make them one argument, so **banner** only produces one line of output.

3-9. SLIDE: The banner Command

Instructor Notes

Purpose

To introduce the `banner` command and, more importantly, the effect arguments have on the output of commands.

Key Points

The double quotation marks around the two words `Hi` and `There` make them appear as one word on the command line because the white space between the words is interpreted as part of the string instead of a delimiter.

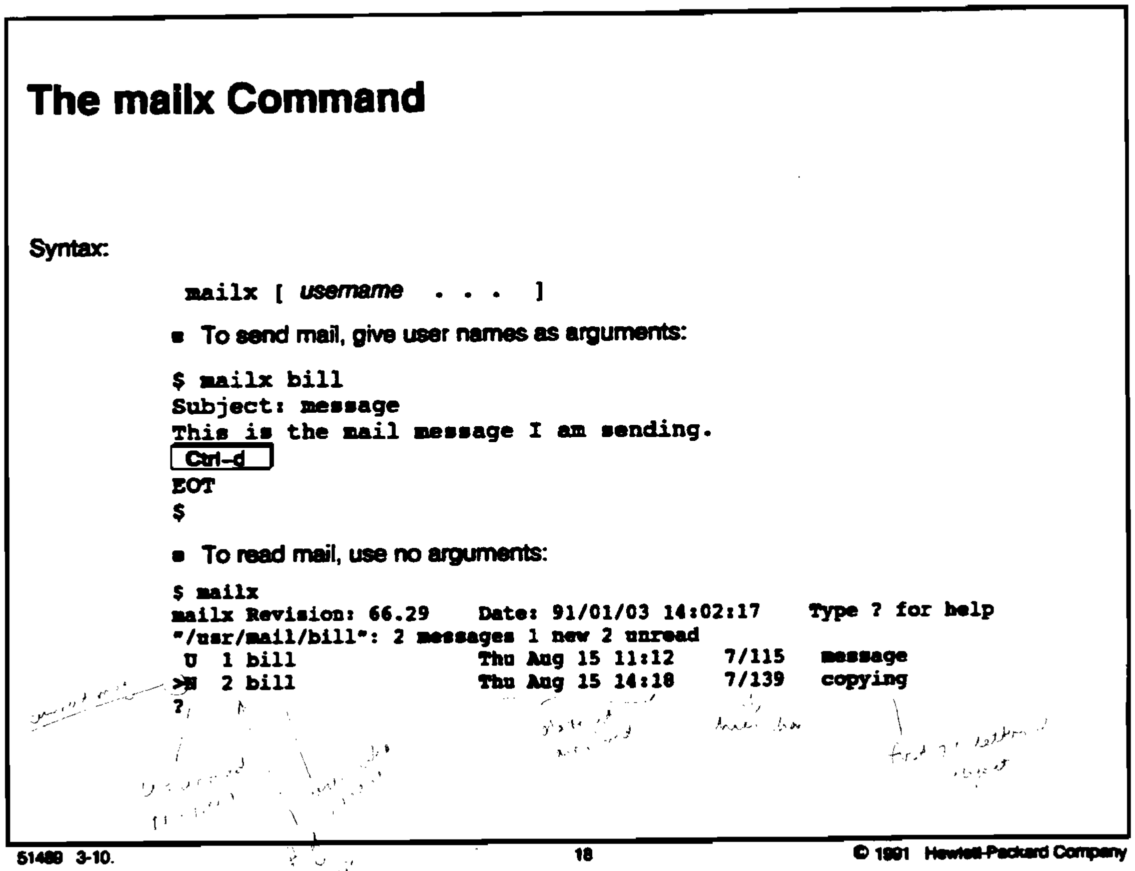
Chalkboard Notes

Give some examples of `banner` commands, arguments, and output.

Transition

We can send and receive messages through the UNIX system's electronic mail facility.

3-10. SLIDE: The mailx Command



The mailx Command

Syntax:

```
mailx [ username . . . ]
```

- To send mail, give user names as arguments:

```
$ mailx bill
Subject: message
This is the mail message I am sending.
Ctrl-d
EOT
$
```

- To read mail, use no arguments:

```
$ mailx
mailx Revision: 66.29    Date: 91/01/03 14:02:17    Type ? for help
"/usr/mail/bill": 2 messages 1 new 2 unread
 U  1 bill               Thu Aug 15 11:12    7/115    message
>N  2 bill               Thu Aug 15 14:18    7/139    copying
?
```

[illegible]

51489 3-10. 18 © 1991 Hewlett-Packard Company

Student Notes

The `mailx` command allows you to send and receive mail electronically on your UNIX system computer.

To send mail to another user on the system, use the `mailx` command and specify a list of users to send your message to. After you press Return, the `mailx` command will wait for you to type in the subject and a message. After your last line, you must press Ctrl-d to send the message. You will get EOT and your shell prompt back.

```
$ mailx bill
Subject: copying
I have some workbooks that I need copied.
See me when you have a chance.
Ctrl-d
EOT
$
```

If Bill wanted to read this message, he would have to type `mailx` with no arguments.

```
$ mailx
mailx Revision: 66.29    Date: 91/01/03 14:02:17    Type ? for help
"/usr/mail/bill": 2 messages 1 new 2 unread
 U  1 bill                Thu Aug 15 11:12    7/115   message
>N  2 bill                Thu Aug 15 14:18    7/139   copying
? 2
Message  2:
From gerry Thu Aug 15 14:18 EDT 1991
To: bill
Subject: copying
Status: R

I have some workbooks that I need copied.
See me when you have a chance.

?
```

The ? is the `mailx` prompt. At this point there are a few commands you can use to manipulate `mailx`:

#	Read the message whose number you have typed.
d#	Delete the message whose number you have typed.
s# *filename*	Save the message whose number you have typed in *filename*.
h	List the headers from each message in your mailbox.
?	Display a help message of `mailx` commands.
q	Leave `mailx`; unread mail will be in your mailbox the next time you read your mail.

[illegible]

[illegible]

3-10. SLIDE: The mailx Command

Instructor Notes

Purpose

To introduce a general purpose electronic mail facility. The `mailx` command provides a more enhanced user interface than `mail`.

Key Points

There are several different mail facilities available on HP-UX. `mailx`, however, is standard on most UNIX systems. It is similar in format to `elm` which is the default mail program on HP VUE.

You may need to mention that if messages are read, but not explicitly saved, they will be saved in a file called `mbox` in the login directory.

Where Problems Arise

Some students may hit (Return) at the ? prompt. Initially, this will read the message pointed to by the arrow. Subsequent (Return)s will read the next message in the list until it reaches the `EOF`.

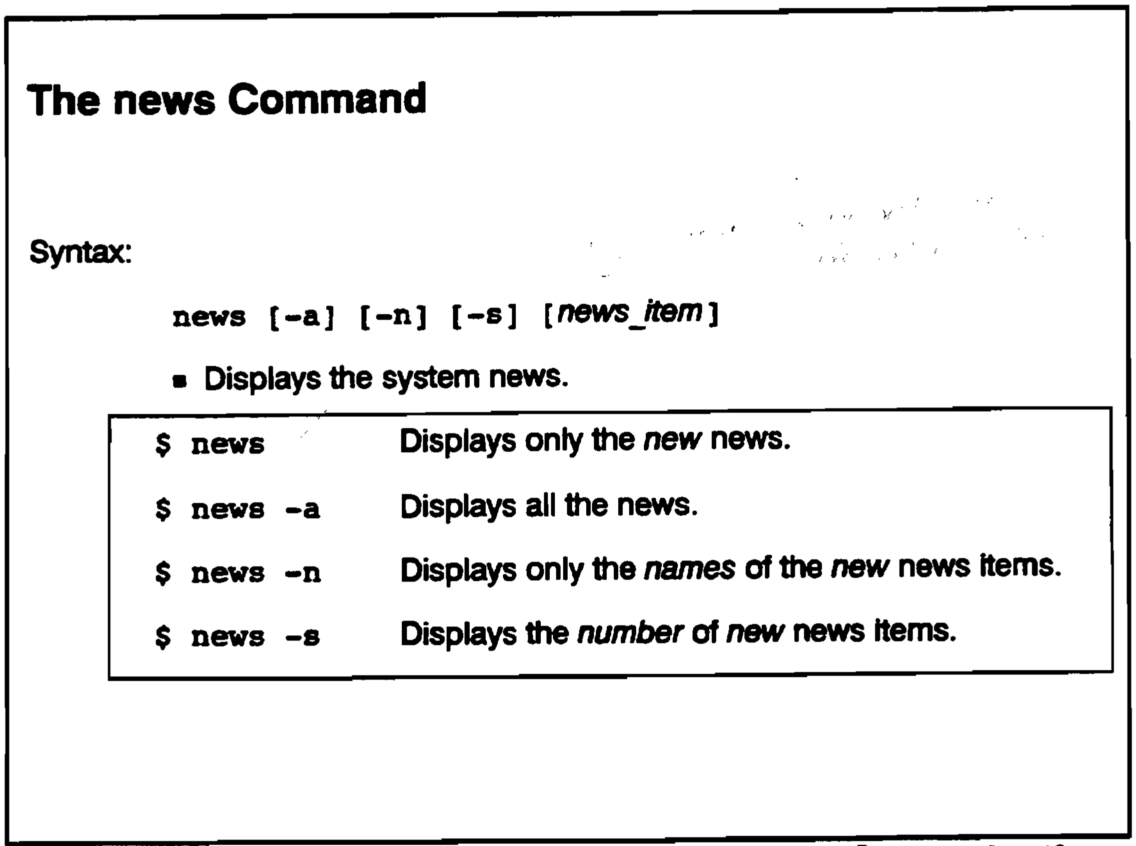
Using `d` and `s` without indicating a message number will perform the designated action on the *current* message.

If students delete messages by mistake, you may want to mention the `x` command. This will allow them to leave `mailx` without updating the system mailbox.

Transition

We can read news sent out by the system administrator.

3-11. SLIDE: The news Command



The news Command

Syntax:

```
news [-a] [-n] [-s] [news_item]
```

- Displays the system news.

`$ news`	Displays only the *new* news.
`$ news -a`	Displays all the news.
`$ news -n`	Displays only the *names* of the *new* news items.
`$ news -s`	Displays the *number* of *new* news items.

51489 3-11. 19 © 1991 Hewlett-Packard Company

Student Notes

The `news` command reports news items and is available to everyone on the system. News items are created by the system administrator to keep users informed of topical events such as new printers, facilities, or games. However, the news items can be on any topic the system administrator desires.

You can see if there is any *new* news and read it by issuing the `news` command.

The options to `news` are:

`-a`	Read all the news there is regardless of the time when it was entered.
`-n`	Report only the *names* of the *new* news items that exist.
`-s`	Find out how many *new* news items there are.

If you use no options, the `news` command will only show you the news that was entered since the last time you read the news. This is done automatically by the news system by creating a file in your login directory called `.news_time`. The date stamp on `.news_time` is the date that was on the system clock the last time you read the news.

If there are a lot of news items, and you wish to read a particular one, use the `news` [*news item*] form of the command.

3-11. SLIDE: The news Command

Instructor Notes

Purpose

To introduce the `news` command.

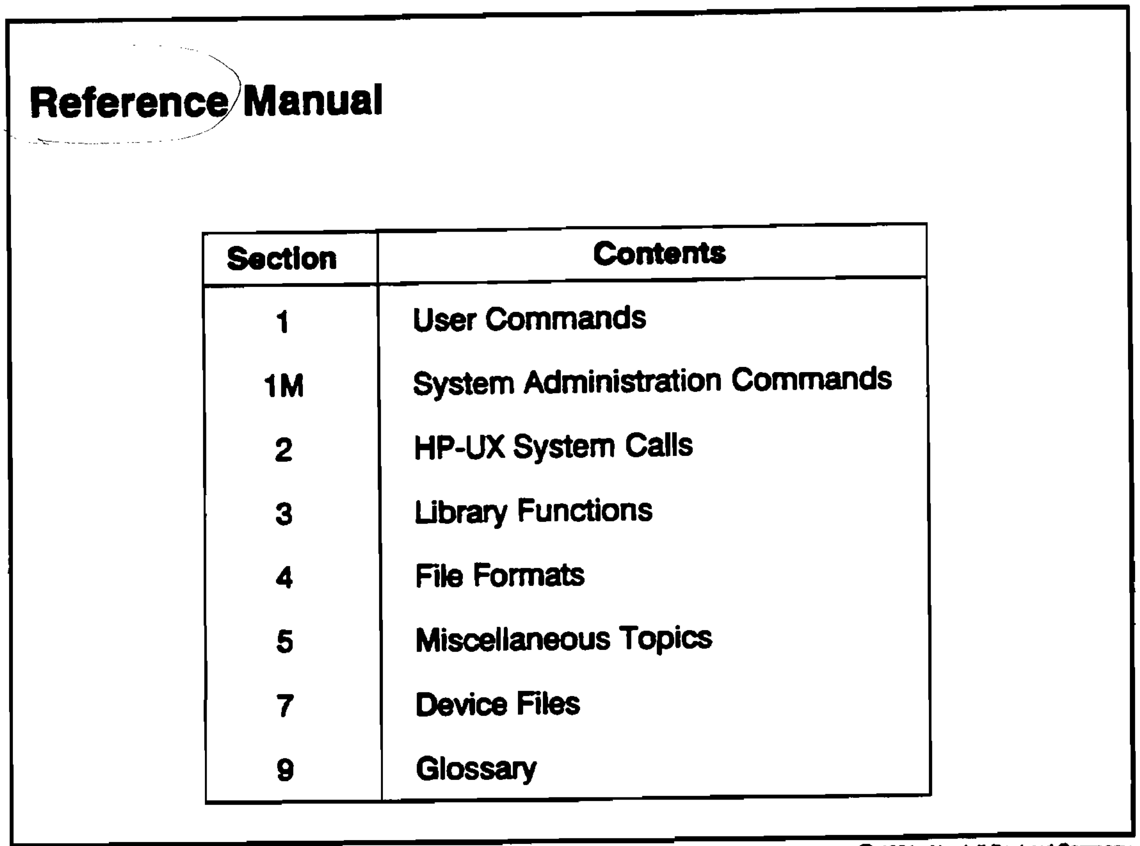
Key Points

Only news items that the user has not read will be printed when the command is entered. If an interrupt is typed during the printing of a news item, the printing of this item will stop and the next one will start. The use of any of the options, `-a`, `-n`, `-s`, does not update the date stamp on `.news_time`.

Transition

Where do we find out more about the commands?

3-12. SLIDE: Reference Manual



Reference Manual

Section	Contents
1	User Commands
1M	System Administration Commands
2	HP-UX System Calls
3	Library Functions
4	File Formats
5	Miscellaneous Topics
7	Device Files
9	Glossary

51489 3-12 20 © 1991 Hewlett-Packard Company

Student Notes

The *UNIX Reference Manual* is very useful for finding the function of a command and the command's usage. However, it is a reference manual and not a tutorial. Thus, it is not used for learning the basics of the UNIX operating system. These manuals are intended for the user who wants to know the details involved in a command and its arguments.

The Hewlett-Packard reference manual is called the *HP-UX Reference Manual.*

The manual is divided into several sections:

- Section 1: User Commands.
- Section 1M: System Administration Commands.
- Section 2: System Calls.
- Section 3: Library Functions.
- Section 4: File Formats.

- Section 5: Miscellaneous Topics.
- Section 7: Device Files.
- Section 9: Glossary.

You may notice that sections 6 and 8 are missing. Section 6 listed games on the original UNIX system; no games are currently supported on HP-UX. Items that were in section 8 have been moved to section 1M. You will find that most UNIX operating system manuals have this format.

We will be dealing primarily with "Section 1: User Commands." If you administer or program your system, you will also be using the other sections.

In the UNIX world, you will often see references to commands given with a number in parentheses. For example, if you ever see: *who(1)* in a document, it is a reference to the manual entry for the *who* command in *section 1* of the reference manual.

Within each section, commands are listed in alphabetical order. There is also a table of contents included at the beginning of each volume containing a complete listing of all manual entries in the order they appear in each section as well as any keywords in the entries. This provides an easy path for locating commands and features whose keyword names are not the same as the title heading on the manual entry.

The permuted index is a keyword listing of the entries in the manual. For example, if you want to find out which command to use to rename files, you would look up "rename" in the permuted index. There it would tell you to use the `mv` command.

Exercise

1. If you have printed copies of the *HP-UX Reference Manual*, try to find the `who` command.

3-12. SLIDE: Reference Manual

Instructor Notes

Purpose

Now that the students have a general understanding of commands, it is time to look at the reference manuals. In addition to explaining the headings provided for each command, also have the students look at the permuted index.

Key Points

The commands are listed in alphabetical order in sections.

The students should get in the habit of referring to the manual when executing a given command to see what other features the command has. Normally, only the basic form of a command will be discussed in class.

Where Problems Arise

The manual is sometimes difficult to understand. The manual is a *reference* guide. It is *not* a tutorial.

Presentation Suggestions

Review the format of the manual (for example, cover the introduction to HP-UX which describes the manual's sections, look at the permuted index, and then look at least one of the section 1 commands, perhaps `date`).

Transition

Now that we know about the reference manual, let's look at the format of an individual page.

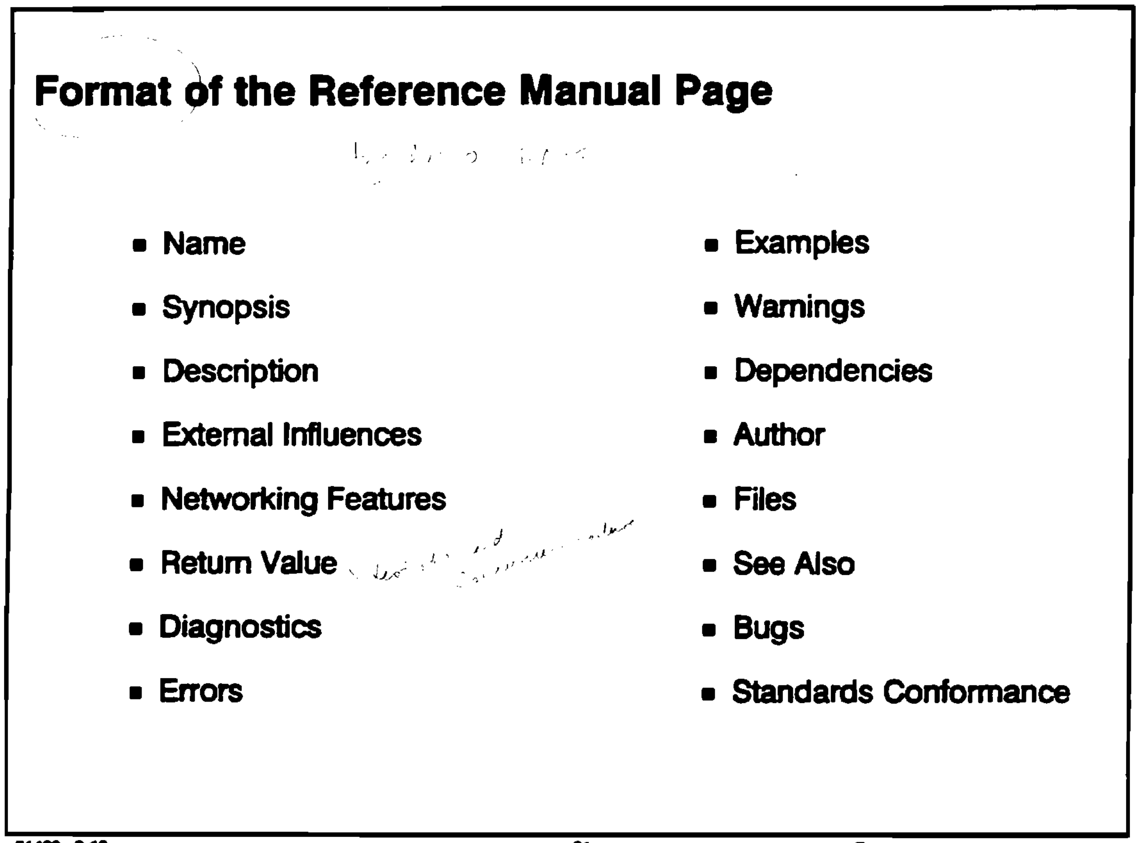
Answer

1. If you have printed copies of the *HP-UX Reference Manual*, try to find the `who` command.

Answer:

Since it is a user command, you would look under who(1).

3-13. SLIDE: Format of the Reference Manual Page



Format of the Reference Manual Page

[illegible]

- Name
- Synopsis
- Description
- External Influences
- Networking Features
- Return Value [illegible]
- Diagnostics
- Errors
- Examples
- Warnings
- Dependencies
- Author
- Files
- See Also
- Bugs
- Standards Conformance

51489 3-13. 21 © 1991 Hewlett-Packard Company

Student Notes

Each manual "page" (some commands take up more than one page) has several major headings. Manual pages do not always have all the headings on them. The following lists each heading and gives a description of what it means:

Name	Contains the name of the command and a brief description.
Synopsis	Indicates how the command is invoked. Items in **boldface** are to be typed at the terminal exactly as shown. Items in square brackets ([ ]) are optional. Items in regular type are to be replaced with appropriate text that you choose. Dots (. . .) are used to show that the previous argument may be repeated. If in doubt about the meaning of the synopsis, read the Description.
Description	Contains a detailed description of what the command does and the meaning of the options and arguments. Options are special arguments that somehow modify the command's function.

External Influences	Information pertaining to programming in various spoken languages.
Networking Features	Information needed if you are using a particular networking feature on your computer.
Return Value	Describes values returned on completion of the program call.
Diagnostics	Diagnostic indications that may be produced.
Errors	Lists error conditions and their corresponding message or return value.
Examples	Some manual pages have examples of usage.
Warnings	This is the same as the Bugs heading.
Hardware Dependencies	Points out variations in HP-UX operation that are related to the use of specific hardware.
Author	Indicates the origin of the software documented by the manual entry.
Files	Lists the file names that the command uses.
See Also	Refers you to other pages in the manual or other documentation for additional information.
Bugs	Used to warn you of potential problems in using the command, occasionally suggesting fixes.
Standards Conformance	Lists the industry standard specifications to which the command conforms.

3-13. SLIDE: Format of the Reference Manual Page

Instructor Notes

Purpose

Knowing about the manual is not enough. Students must also know how to read a manual page and understand what each of the entries means.

Key Points

These are the meanings of the headings on each manual page.

Teaching Question

Ask the students to locate the `date` command and explain the headings included in its manual page.

Where Problems Arise

The manual is often difficult to understand. The manual is a *reference* guide. It is *not* a tutorial.

Evaluation Questions

Ask the students to find the manual page for the `who` command and explain the headings.

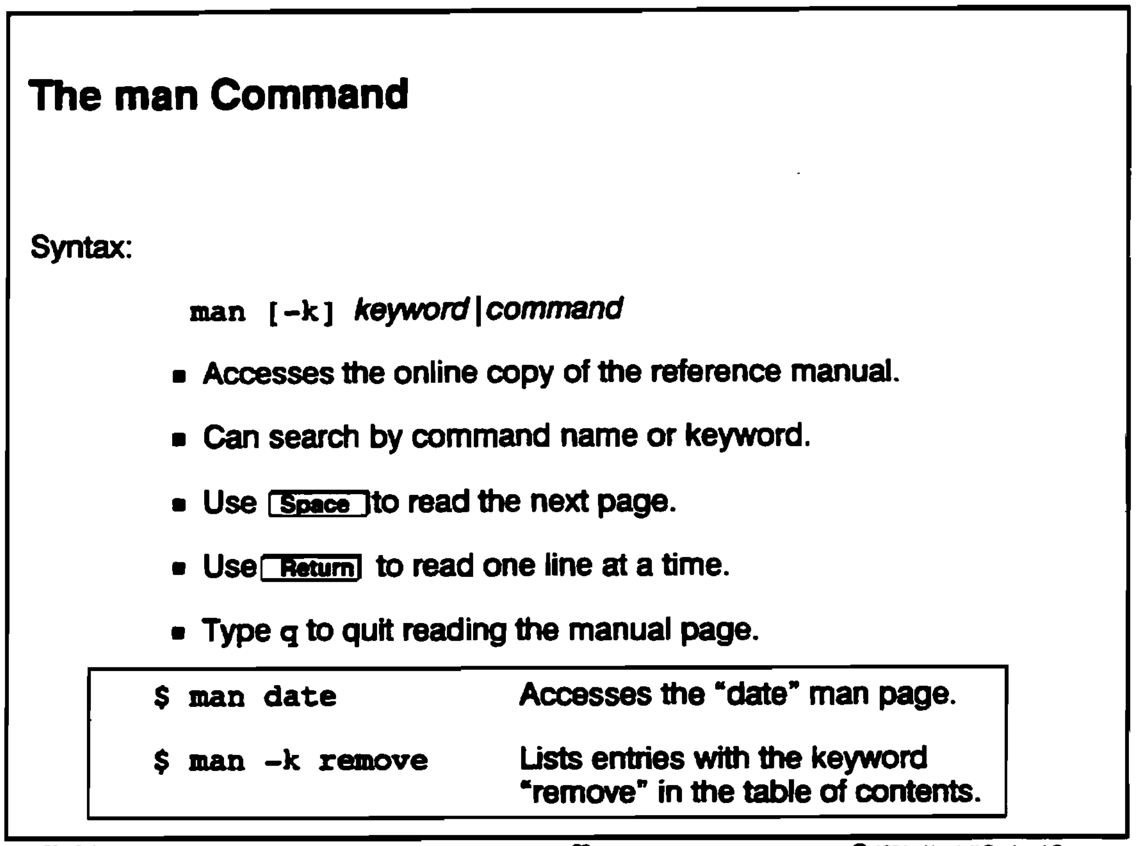
Presentation Suggestions

Use the manual to illustrate each point on the slide.

Transition

Now let's look at a how to access the reference manual that is on the system.

3-14. SLIDE: The man Command



The man Command

Syntax:

`man [-k] `*keyword|command*

- Accesses the online copy of the reference manual.
- Can search by command name or keyword.
- Use [Space] to read the next page.
- Use [Return] to read one line at a time.
- Type q to quit reading the manual page.

`$ man date`	Accesses the "date" man page.
`$ man -k remove`	Lists entries with the keyword "remove" in the table of contents.

51489 3-14. 22 © 1991 Hewlett-Packard Company

Student Notes

The HP-UX manual is stored online. If you do not have a manual nearby and need to look something up, you can use the man command. It will access the information and display the manual page on your screen.

To access the online reference manual, you can use the man command:

```
$ man command_name (Return)
```

where *command_name* is the name of a UNIX system command. The man command will display the manual page for the specified command, one screen at a time. To see the next page of text, press the space bar. Pressing the (Return) key will advance the text on your screen one line at a time.

The manual page will stop if you read the whole thing or type q.

When `man` is used with the `-k` option followed by a keyword, it will print a one-line summary of each manual entry whose listing in the table of contents contains that keyword. Your system administrator must set up your system before you can use the `-k` option.

Exercises

1. How would you find more information on `man` using the `man` command?

2. Try to get information on the `date` command using `man`.

3-14. SLIDE: The man Command

Instructor Notes

Purpose

Students can get help from the computer if they can remember this command.

Key Points

Not all systems have the online reference manual loaded. Some sites may take it off the system because of all the disk space it occupies.

The `-k` option is from Berkeley. It is similar to searching through the permuted index.

Presentation Suggestions

Discuss the `-k` option to `man`.

Transition

Now that you know some of the basics of HP-UX, you can try some exercises.

Answers

1. How would you find more information on `man` using the `man` command?

Answer:

```
$ man man
```

2. Try to get information on the `date` command using `man`.

Answer:

```
$ man date
```

3-15. LAB: Exercises

Each person should do these exercises once. If you have a partner, each of you should do each exercise at the terminal.

1. Practice logging off. Then log back in again.

2. Run the `hello` program that is in your home directory.

3. Look in the reference manual to find out which `who` option is used to print column headings above the output. Try `who` using this option.

4. Using the `echo` command, display `HP-UX is fun` on your screen. Try this again, making sure the words are three spaces apart.

5. Using the `banner` command, display the words `Hi There` in large letters on your screen. Try this again, making sure both words are displayed on the same line.

6. Send a mail message to your partner. Read the mail that they sent to you. Now save that mail message to a file called mail.save. What command did you use?

7. Read the system's news. What command did you use?

[illegible]

8. How can you make the date command display the date in mm/dd/yy format?

date [illegible]

[illegible]		
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]
cd	[illegible]	lp
[illegible]	pwd	chgrp
cat	ls	rmdir
mkdir	[illegible]	[illegible]

[illegible]

3-15. LAB: Exercises

Purpose

To provide the first hands-on experience in logging in and off and executing some simple commands.

Lab Setup

Prior to starting this lab, the instructor should assign mail partners and create a news item.

Transition

Now that we know how to log in to the system and execute commands we can look at some file commands.

Solutions

Each person should do these exercises once. If you have a partner, each of you should do each exercise at the terminal.

1. Practice logging off. Then log back in again.

Answer:

$ Ctrl-d Return or `$ exit` Return

2. Run the `hello` program that is in your home directory.

Answer:

```
$ hello
```

3. Look in the reference manual to find out which `who` option is used to print column headings above the output. Try `who` using this option.

Answer:

```
$ who -H
NAME      LINE      TIME
gerry     tty0p2    May 25 10:06
jimd      tty1p0    May 25 13:10
doug      tty1p1    May 25 08:04
frank     tty1p2    May 24 08:29
```

4. Using the `echo` command, display `HP-UX is fun` on your screen. Try this again, making sure the words are three spaces apart.

Answer:

```
$ echo HP-UX is fun
HP-UX is fun

$ echo "HP-UX   is   fun"
HP-UX   is   fun
```

5. Using the banner command, display the words Hi There in large letters on your screen. Try this again, making sure both words are displayed on the same line.

Answer:

```
$ banner Hi There
#     #
#     #     #
#     #     #
#######     #
#     #     #
#     #     #
#     #     #

#######
   #     #    #  ######  #####  ######
   #     #    #  #       #    # #
   #     ######  #####   #    # #####
   #     #    #  #       #####  #
   #     #    #  #       #   #  #
   #     #    #  ######  #    # ######

$ banner "Hi There"
#     #                #######
#     #     #             #     #    #  ######  #####   ######
#     #     #             #     #    #  #       #   #   #
#######     #             #     ######  #####   #   #   #####
#     #     #             #     #    #  #       #####   #
#     #     #             #     #    #  #       #   #   #
#     #     #             #     #    #  ######  #    #  #######
```

6. Send a mail message to your partner. Read the mail that they sent to you. Now save that mail message to a file called mail.save. What command did you use?

Answer:

```
? s1 mail.save
```

7. Read the system's news. What command did you use?

Answer:

```
$ news
```

Advanced:

8. How can you make the date command display the date in mm/dd/yy format?

Answer:

```
$ date +%D

OR

$ date +%m/%d/%y
```

Module 4 — Directories and Files

Objectives

Upon completion of this module, you will be able to do the following:

- Understand the UNIX system file tree structure.
- Explain the role of directories.
- Explain the difference between an absolute path name and a relative path name.
- Use the `ls`, `cd`, and `pwd` commands to traverse the directory structure.
- Understand how to organize related files.

[illegible]

[illegible]

[illegible]

[illegible]

• No real restriction [illegible] file names [illegible]

[illegible]

choose a - z

A - Z

0 - 9

. _ underscore

+ -

[illegible]

Module 4 — Directories and Files

Overview of Module 4

This module will clarify the structure of the file system and how users move around within it.

This module may take from 1 to 2 hours to complete, including the terminal exercises.

Motivation

It is an absolute *must* that users know and understand the file system so that they can utilize their UNIX system effectively.

Exercises

The files/directories used in these exercises should be installed from the lab tape into each student's home directory.

Exercise Files

Students will be using several files and the `sample.tree` directory.

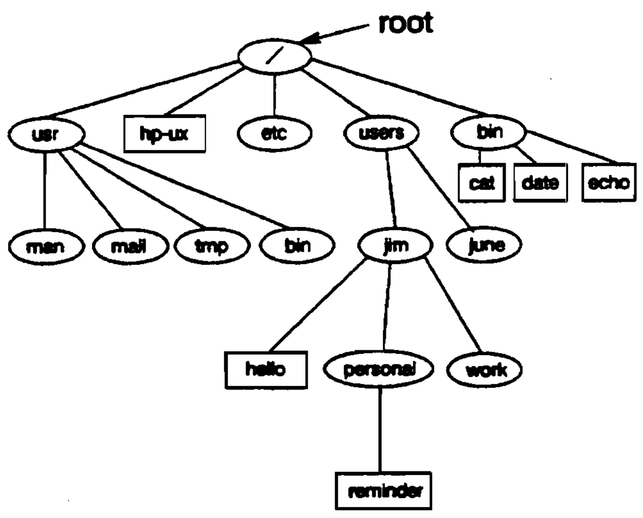
Module Transition

This module introduces a "working" understanding of the file system structure. The following modules will show other commands you can use to manipulate files within this directory structure.

4-1. SLIDE: UNIX File System

UNIX File System

A File System Tree



- The UNIX file system is hierarchical.
- The top is called the "root."
- A directory is a way of organizing the files.
- Directories can contain other subdirectories.

⬭ = directories

▭ = files

51489 4-1. 23 © 1991 Hewlett-Packard Company

Student Notes

The UNIX operating system makes it very easy to manipulate files. In fact, most of the activity on a UNIX system centers on file management. As a user, you will create many files. The files will hold text and programs and will serve a variety of purposes. For example, you might be involved in a software development project. You will need files to hold, among other things, specifications, user documentation, and programs. If you are in a computer-aided design (CAD) environment, you will be creating and using drawing and documentation files. Since you will be working with files, you will need a way to organize them.

Organizing your files takes some thought. This is similar to the way you might organize your personal expenses. You might have basic categories of Household, Clothes, Transportation, and so on. Under Household you may have categories such as Gas, Electric, Food, and Repairs. Under each of these you may have still more subcategories.

The UNIX operating system provides an analogous mechanism for organizing files called **directories**. Directories contain information about files and other directories. In the example above, Household, Clothes, and Transportation, would be directories. The actual receipts, notes, and canceled checks would be the files.

In this module we will look at directories: how to use directories and how to move from one directory to another.

There are several important points to note about directories:

- The file system is organized hierarchically.
- There is one and only one "top" of the hierarchy. This type of organization is called a **tree**.
- The top of the tree is called the **root** (the tree is upside down). In the UNIX operating system, the root of the tree is named "/", which is called a "slash."
- "/" is a directory; it contains files and more directories.
- Each of the directories under "/" may contain other files or directories or both.
- Only directories can contain other files and other directories.

4-1. SLIDE: UNIX File System

Instructor Notes

Purpose

This is the starting point for establishing an understanding of the UNIX hierarchical file system.

Key Points

Similar to an organization chart in its hierarchy, the file system provides a "boss" called "root" at the top. Similarly, at the bottom (as well as anywhere in the middle) exist "files" as "employees who have no one reporting to them." The middle management would be subdirectories.

Teaching Question

Ask how directories might be used to organize files.

Positive Instance

There has to be some method of logically establishing and tracking files. The UNIX operating system uses directories.

Negative Instance

Without directories there would be no way to maintain any formal method of tracking files. Finding files would take a long time.

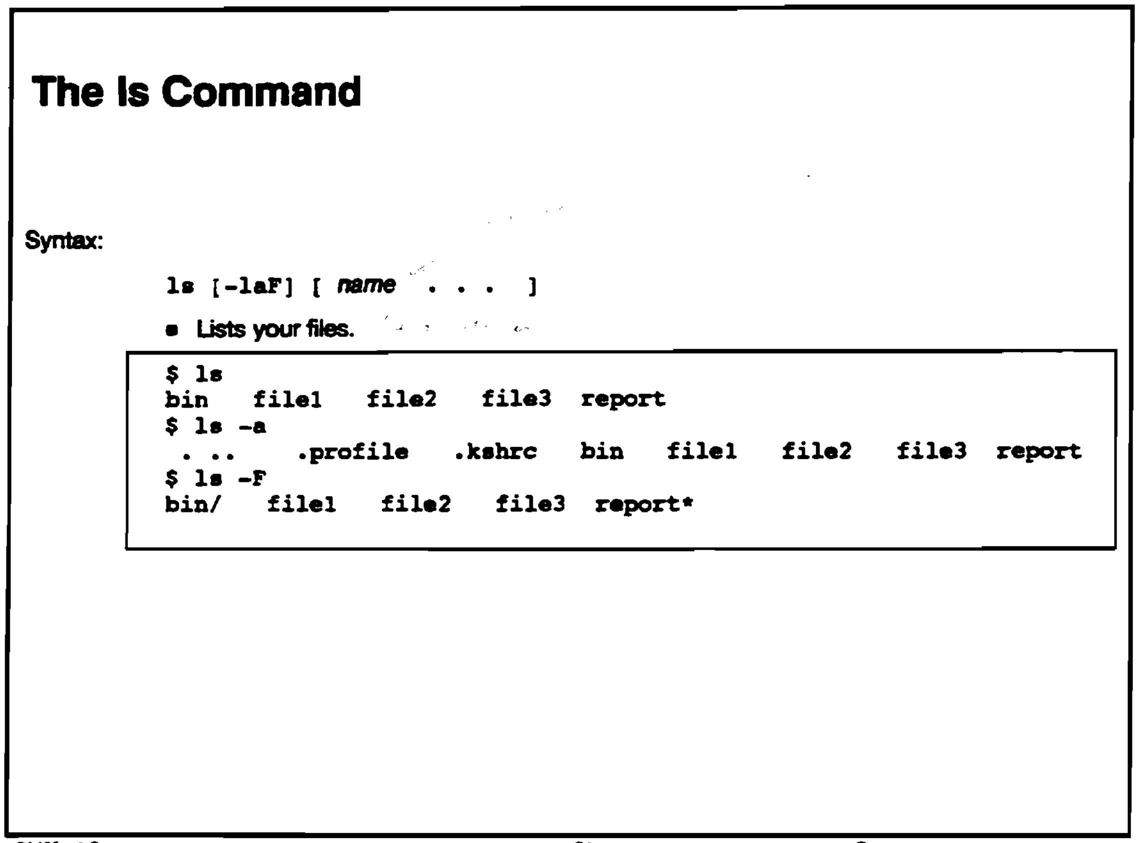
Slide Notes

Show and explain the slide.

Transition

We can use the `ls` command to display the contents of a directory on our screen.

4-2. SLIDE: The ls Command



The ls Command

Syntax:

```
ls [-laF] [ name . . . ]
```

- Lists your files.

```
$ ls
bin   file1   file2   file3  report
$ ls -a
 . ..     .profile   .kshrc   bin   file1   file2   file3  report
$ ls -F
bin/   file1   file2   file3  report*
```

51489 4-2. 24 © 1991 Hewlett-Packard Company

Student Notes

The `ls` command is used to list the contents of your current directory (like looking into an open file cabinet drawer). These are the files and directories that are currently available for you to use. The `ls` command is one of the most frequently used UNIX system commands.

If the `ls` command is given no arguments, it will list all files except for those that begin with a dot (.). If it is given arguments, it will list files according to its arguments.

Some of the common `ls` options are the following:

`-a`	List all entries (including dot files).
`-F`	List in full format (put a "/" after directory names and a "*" after program file names).
`-l`	List in long format, showing more information about the files.

Files with names that start with a dot (.) are called **dot files**. Dot files are useful for storing information or programs that do not change very often and that you do not normally want cluttering up the `ls` output. The `ls` command normally does not report dot files. In order to see dot files, use `ls -a`. We will discuss some important dot files in Module 11.

To get a listing that shows which names in the listing are files, directories, or programs, we can use the `-F` option. This option puts a "/" after directory names and a "*" after program names.

The `-l` option is also very useful. It provides a long listing of the current directory contents. We will see more about the `ls -l` command in Module 6.

This slide shows a fragment of a terminal session. In order to conserve space on the slides, we will most often only show a part of a terminal session. The last shell prompt indicates that the shell is ready to accept more input and the terminal session could continue.

Each of these files on the slide already existed. Each one has a name and associated information. Each one is stored on disk.

Exercises

1. Try the `ls` command. You should see the files in your current directory. How does the output of the `ls -F` command differ from that of `ls`? Try both commands.

2. How would you obtain a full listing of the /tmp directory?

4-2. SLIDE: The ls Command

Instructor Notes

Purpose

As is the case in all commands introduced in this manner, this illustration of the `ls` command shows its more frequently used options. Also note that [args] are optional file names (or path names, which will be explained in detail later in this course).

This is the initial slide indicating the manner in which most illustrations will be shown: Abbreviated sessions will be shown to show specific examples of what is being covered at the time. This example also shows heavy use of file naming conventions, which is normal in software and document production environments.

Key Points

The `ls` command lists the contents (file names) of directories as specified in [args]. The default output from an `ls` command is a listing of file names in the current directory. The `ls -F` command (which is an option developed by Berkeley) and the `ls -l` command are used so often that HP has developed abbreviations for them. Typing `lsf` is equivalent to typing `ls -F`; typing `ll` is equivalent to typing `ls -l`.

The output format of `ls` differs with different versions of the UNIX operating system. The Bell versions lists the files vertically in a single column unless you use the `-C` option. The Berkeley version lists the files in a multi-column format.

Where Problems Arise

Some students may initially have difficulty understanding file names having the a digit as the first character or having dots (.) in their names. To alleviate any of their lingering doubts, just remind them that most ASCII characters may be used in file names even though it is still advisable to avoid special characters other than dots (.), plus signs (+), and underscores (_). We will see more on files in Module 5.

Presentation Suggestions

Use "reveal" to show the command line, then the output, and lastly the shell prompt. Note that it is generally good practice to get the students "seeing" the distinct parts of command execution (in other words, the command line first and then its output).

Transition

The first directory to look at is the user's login directory.

Answers

1. Try the `ls` command. You should see the files in your current directory. How does the output of the `ls -F` command differ from that of `ls`? Try both commands.

Answer:

`ls -F` displays a "/" after directory names and a "*" after program names.

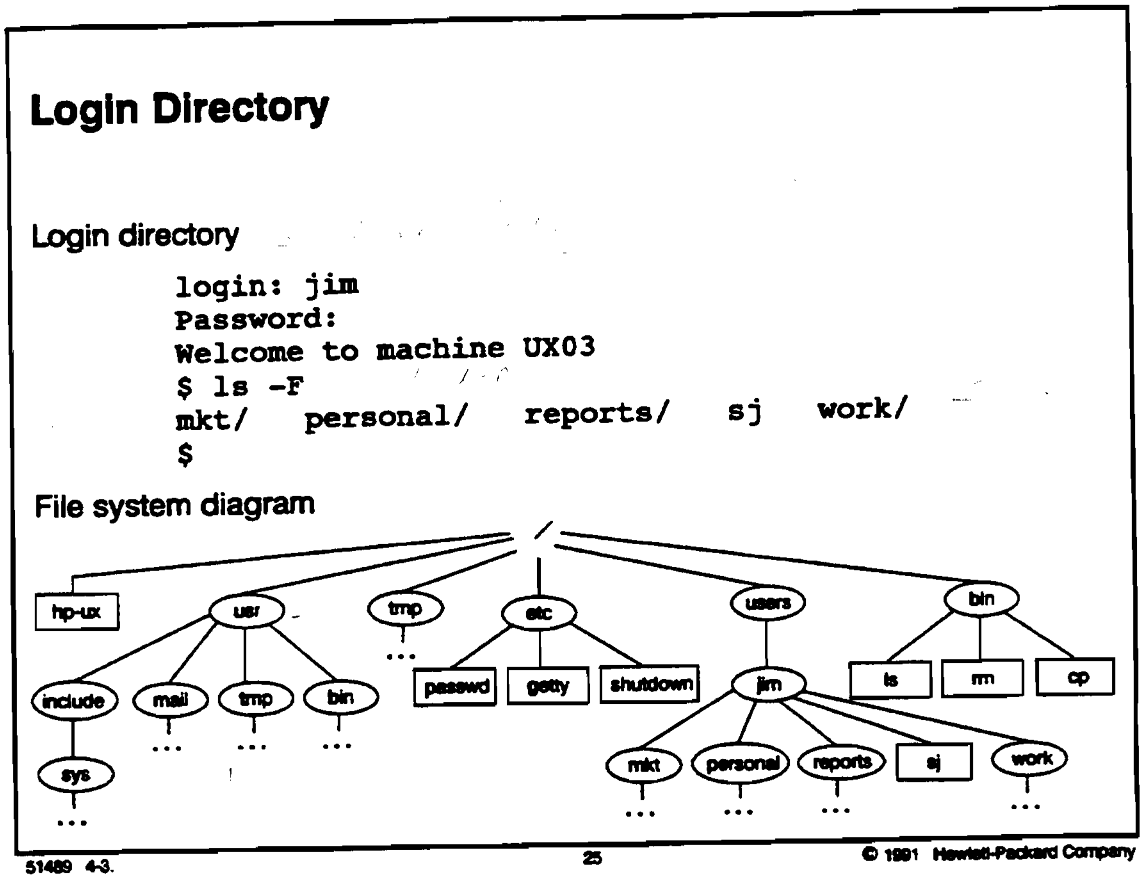
2. How would you obtain a full listing of the /tmp directory?

Answer:

Provide /tmp as an argument to the `ls -F` command:

```
$ ls -F /tmp
./    ../    .X11-unix/    data    junk    reconfig.log
update.log
$
```

4-3. SLIDE: Login Directory



Login Directory

Login directory

```
login: jim
Password:
Welcome to machine UX03
$ ls -F
mkt/   personal/   reports/   sj   work/
$
```

File system diagram

/
hp-ux
usr
tmp
etc
users
bin
include
mail
tmp
bin
passwd
getty
shutdown
jim
ls
rm
cp
sys
mkt
personal
reports
sj
work

51489 4-3. 25 © 1991 Hewlett-Packard Company

Student Notes

Recall that UNIX directories are places where you sit and look around. You are always "in" a directory. That is, whenever you are on a UNIX system, you always have a current directory. That is the directory that is available to you at that moment. Continuing the personal expense analogy, a file cabinet drawer that is labeled "Household" and that contains folders marked "Gas", "Electric", and so on would be a directory. If you had that file drawer open and were looking at the folders inside, you would be "in" that "directory." The UNIX operating system keeps track of your current directory.

A user has one special directory for which he or she is responsible. This directory is called the **login directory** or the **home directory.** When a user logs in, the shell sets the current directory to be the login directory. This gives the user a known starting point. This is a directory where a user can organize his or her own files and directories.

[illegible]

4-3. SLIDE: Login Directory

Purpose

To introduce the concept of the current directory.

Key Points

The login directory is the logical place to initially look because it is where the user is "placed" at login and is the current directory. Notice that the login directory can contain files, subdirectories, and programs.

Teaching Question

Ask whether the file names listed in the output of the `ls -F` command are "regular" files or "directory" files.

Presentation Suggestions

Again, use "reveal" to help illustrate the example. You may not want to show the picture of the truncated file system until you have asked the teaching question.

Slide Notes

Point out (by using a slide pen) that the "same" file name can exist in more than one directory.

Transition

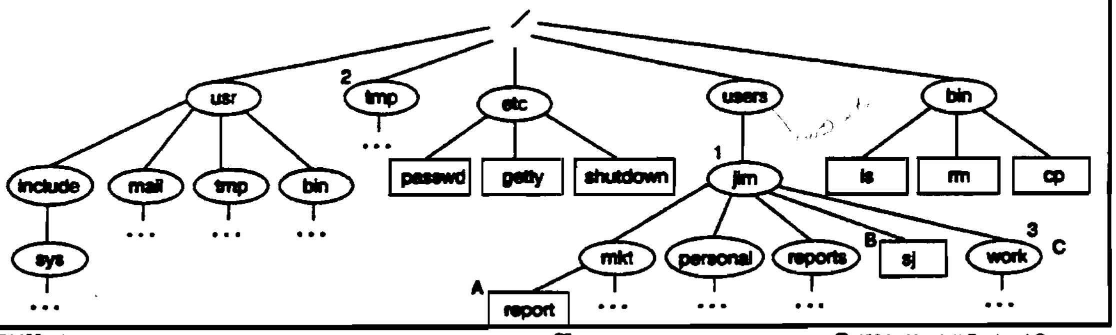
Path names are how we identify files and directories in the UNIX operating system.

4-4. SLIDE: Path Names

Path Names

- Each file and directory are identified by their path names.
- Absolute path names give the full path from the "root" directory.
- Relative path names give the path from the current directory.

Absolute:	Relative to `/users/jim`:
1. `jim` is `/users/jim`	A. `mkt/report`
2. `tmp` is `/tmp`	B. `sj`
3. `work` is `/users/jim/work`	C. `work`



51489 4-4. 28 © 1991 Hewlett-Packard Company

Student Notes

We define the term **path** to mean some route along the branches of the tree from a directory to some other file or directory.

To visualize a path, put a pencil on the picture of the tree at the starting directory and trace along the branches and through the directories along the way until you reach the desired file or directory. You must move the pencil along the lines, and the pencil tip can never leave the paper.

By "path name," we mean a list of the directories that the pencil point touches on its way from the starting directory to the destination file or directory. Note that given a starting point and an ending point, there is only *one* path to take. In particular, note that any file or directory in the tree has a unique path to it from "/". This is a very important rule in the UNIX file system.

Path names can be used wherever file names or directory names can be used. We will see examples of this in the following modules.

There are two kinds of path names: absolute and relative.

Absolute path names specify where a file or directory may be found relative to the root directory ("/"). Absolute path names are written by starting with a "/", which tells the UNIX operating system that you are specifying a path from the root, and then listing all of the directories that the pencil point would touch if you were tracing the path on a picture of the tree. An example of an absolute path name is `/users/jim/mkt`. This path name tells the UNIX operating system to start at "/" (the first "/" in the path name) and look for a directory named `users`. Then in the users directory it looks for a directory named `jim`. Then in the jim directory it looks for a file or directory named `mkt`. The first "/" specifies the root directory. The other "/"s just serve to separate the components of the path name.

Relative path names specify where a file or directory may be found relative to the current directory. Relative path names do not start with a "/". An example of a relative path name is `mkt/prop0`. That tells the UNIX operating system to look for a directory named `mkt` under the current directory, and then in the `mkt` directory to look for a file or directory named `prop0`.

The UNIX operating system allows the use of relative path names only as a typing convenience.

4-4. SLIDE: Path Names

Instructor Notes

Purpose

To explain how to reference *any* file or directory in the file system by specifying its path name.

Key Points

Every file and directory has only *one* "location" in the file system. Its uniqueness is ensured because there is only one path from the root ("/") and there is only one root directory.

Emphasize that *path names* can be used as arguments to commands that operate on files or directories. Most of the time we will use names relative to the current working directory (that is, just file names).

Absolute path names start with "/". Relative path names never start with "/".

Teaching Question

Can we have two files with the same name if they exist in different directories?

Answer: Sure, their absolute path names would be different if they resided in different directories.

Positive Instance

You cannot create two files in the same directory with the same name.

Where Problems Arise

A common perception problem is that a user must be "in" the directory in order to reference files or directories in that directory. That is not true, as we will see shortly.

Presentation Suggestions

Make an analogy to full names versus first names. The name "John Fitzgerald Kennedy" would be a full identifier, while the name "John" would only be accurate within the proper context.

If needed, pick a current directory on the slide and ask the students the relative path names to other directories or files.

Slide Notes

Be sure to point out the slashes in the path names.

Make sure you go over the examples on the slide.

Transition

Knowing that we "arrive" at our login directory at login time and that we can use path names to reference files and directories other than in our login directory, how can we "relocate" to a different directory?

4-5. SLIDE: The cd Command

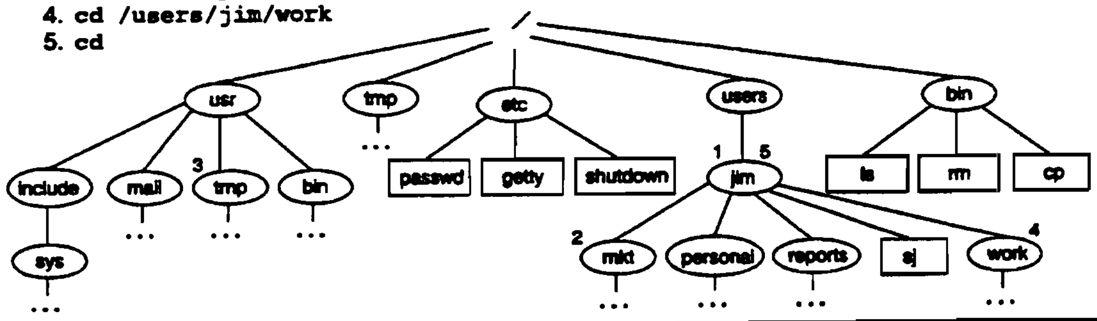
The cd Command

Syntax:

```
cd [ directory ]
```

- Move from current directory to directory specified.
- The only way to change directories is with the `cd` command.
- `cd` with no argument changes to your login directory.

1. start at login directory – /users/jim
2. `cd mkt`
3. `cd /usr/tmp`
4. `cd /users/jim/work`
5. `cd`



51489 4-5. 27 © 1991 Hewlett-Packard Company

Student Notes

We have seen a number of directories, and we know how to specify them using relative or absolute path names. We have also seen that we start out in a directory called the login directory.

The UNIX operating system allows us to use just the file name when we are referencing a file in the current directory. The main reason for changing directories, then, is to get into the directory that contains the files that we want to do work on. This will save typing when we start doing tasks such as copying, removing, and editing files. If we could not change our current directory, we would have to use absolute path names or very long relative path names to specify which files and directories we want to work with.

If invoked with no arguments, `cd` takes you to your login directory. If you give `cd` an argument, which must be a directory, you will move from your current directory to the directory you specified. There can be only one directory argument. (How can you be in two places at once?) You cannot change directories to a file, only to another directory. Naturally, the argument to the `cd` command may be specified as either a relative path name or an absolute path name; the only requirement is that the argument must name a valid directory.

4-5. SLIDE: The cd Command

Instructor Notes

Purpose

There is only one way to relocate (change) to another directory.

Key Points

The reason we want to change to a different directory is to perform some actions on the contents of that directory. After changing directories we can use just the names of the files and not have to use path names.

Teaching Question

Ask how to invoke the `cd` command to change to the root directory.

Answer: `cd /`

Where Problems Arise

The most common error made at this point in their learning is their entering incorrect path name notation.

Evaluation Questions

Ask whether it is necessary to change directories to have access to files or directories in a different directory.

On the slide, ask if you could "cd" **`/users/jim/sj`**. No you cannot because **`sj`** is a file, not a directory.

Presentation Suggestions

Combine the path name knowledge that the students have just acquired to illustrate how `cd` works.

Presentation Suggestions

Make sure you show and explain the slide.

Chalkboard Notes

Board examples of `cd` command lines using relative and absolute path names help a great deal.

Transition

If we change directories several times, it is easy to lose track of where we are. The `pwd` command will tell us what directory we are in.

4-6. SLIDE: The pwd Command

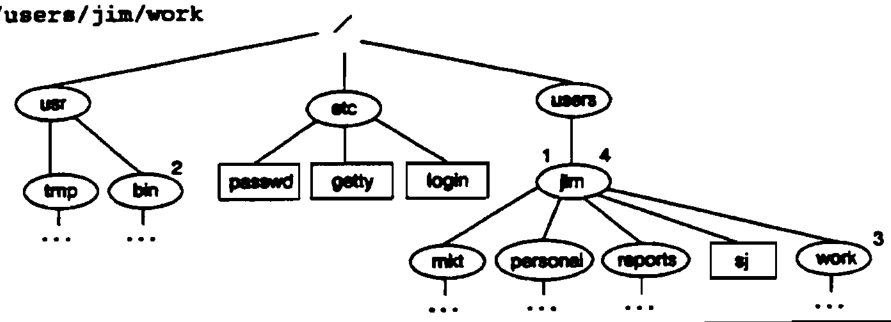
The pwd Command

Syntax:

pwd

- Prints absolute path from root to current working directory.

```
1. $ pwd                              3. $ pwd
   /users/jim                            /users/jim/work
   $ cd /usr/bin                      4. $ cd
2. $ pwd                                 $ pwd
   /usr/bin                              /users/jim
   $ cd /users/jim/work
```



51489 4-6. 28 © 1991 Hewlett-Packard Company

Student Notes

The pwd command is used to display the name of the directory you are in. It reports the *absolute* path name to your current directory.

Do not get discouraged if you get lost while moving around the file system. As you explore the directory structure, you will become comfortable with it and you will learn your way around. It is a little like moving to a new town. At first you need maps and reassurances, but after a while you do not have to think about where you are going.

If you do get lost, cd with no arguments will take you back to your login directory. It is not surprising that a synonym for the login directory is the "home" directory.

4-6. SLIDE: The pwd Command

Instructor Notes

Purpose

To see the absolute path name of the current directory.

Key Points

pwd only prints the absolute path name of the current directory.

Evaluation Questions

Is the path name that is reported the absolute or relative path name?

Slide Notes

Show and explain the slide.

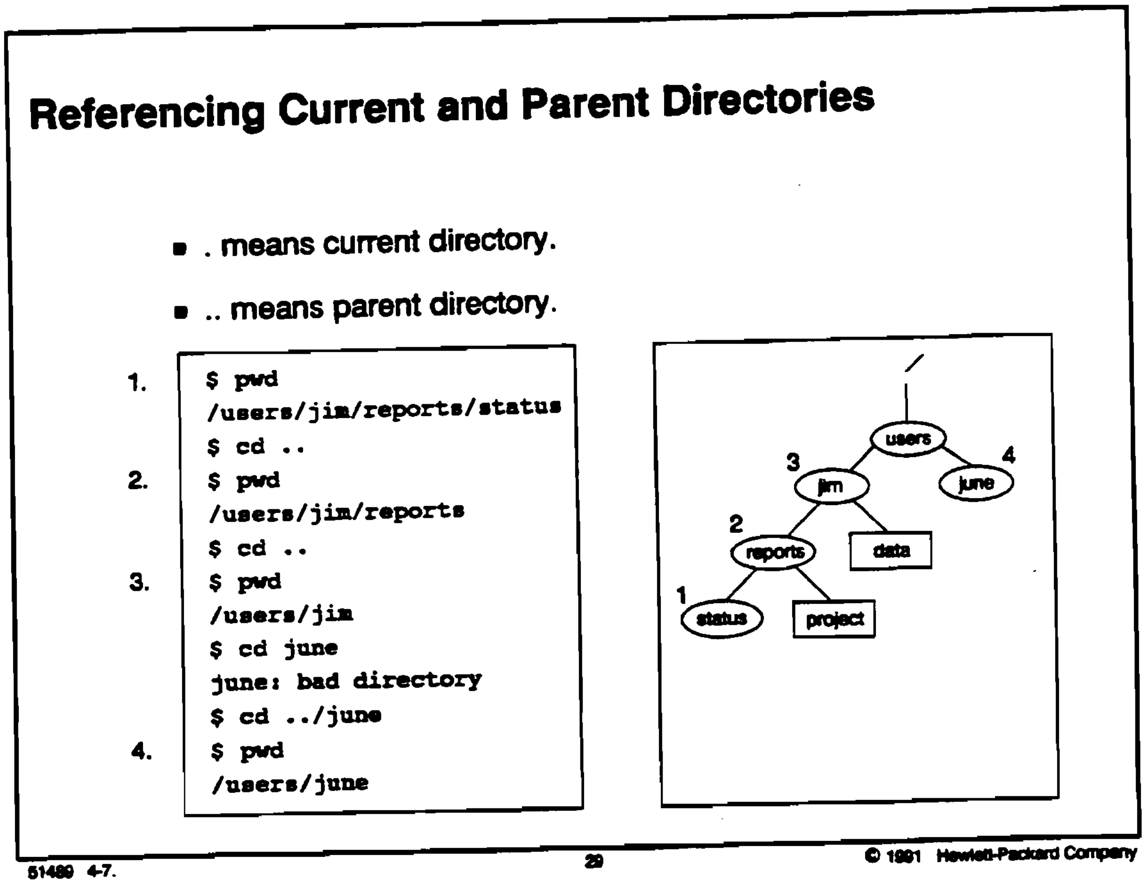
Chalkboard Notes

Go over the exercises on the board if necessary. Drawing pictures usually helps a great deal.

Transition

Now we need to discuss referencing current and parent directories.

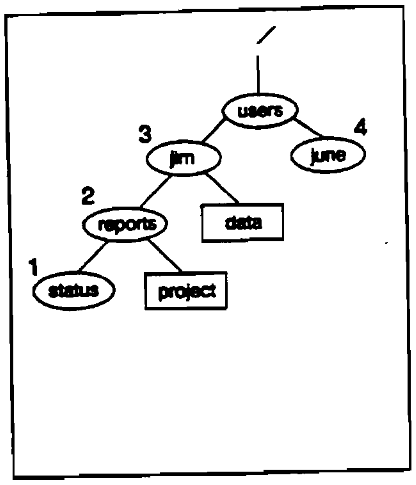
4-7. SLIDE: Referencing Current and Parent Directories



Referencing Current and Parent Directories

- . means current directory.
- .. means parent directory.

```
1.  $ pwd
    /users/jim/reports/status
    $ cd ..
2.  $ pwd
    /users/jim/reports
    $ cd ..
3.  $ pwd
    /users/jim
    $ cd june
    june: bad directory
    $ cd ../june
4.  $ pwd
    /users/june
```



51489 4-7. 29 © 1991 Hewlett-Packard Company

Student Notes

Part of the information that a directory contains is the names of all the files and directories contained in it. The directory is the only place that this file information is kept; nothing else in the entire directory structure has any idea what files or directories are in a particular directory. That is why you cannot skip over directories in path names.

If you want to look at things down the tree, you can specify a relative or absolute path name to the desired directory. How, though, can we specify something up the tree? For instance, how can we list the names of the files in the directory above our current directory? We could, of course, specify the absolute path name to the parent, but the UNIX operating system provides an easier way.

Each directory contains a special name for its parent directory (remember that each directory has only one parent). The directory knows the name of its parent as "dot dot" (..). We can use ".." in relative path names:

```
$ ls -F ..
```

This will give you a full listing of the files in the parent directory. Notice that the directory you are in is included in the list.

We can also use the ".." with path names in the cd command. For instance,

```
$ cd ..
```

takes us up the directory tree, one level closer to the root directory. Specifying actual names takes us down the tree. We can combine ".." with other paths to go up the tree and then back down. For example,

```
$ cd /usr/man
$ cd ../bin
$ pwd
/usr/bin
$
```

To be complete, how can we specify the current directory? We cannot just use its name, since we have already said that when we specify a name, it is understood that we are naming something below the current directory. We use a special name to refer to the current directory. That name is "dot" (.). It is not used as often as "dot dot" (..), but it does have some uses. We will discuss these uses as they arise.

About the slide: We show the effects of using "cd .." by displaying the new current directory at each stage. When we try to cd to "june" we get the `bad directory` message, which tells us that "june" is not contained in the current directory. Instead, we can combine ".." into the relative path name to back up one level and then move back down to june.

4-7. SLIDE: Referencing Current and Parent Directories

Instructor Notes

Purpose

To see how the current directory and its parent are referenced as arguments to commands.

Key Points

To this point, all references to files or directories have been below the current directory or below root. We can also make references up the tree using relative path names. Dot (.) is always the "name" of the current directory (that is, that is how the current directory knows itself). Similarly, the current directory always knows its parent as dot dot (..).

Slide Notes

Show and explain the slide. It is best to keep a pointer on the slide to show the new directory after each move. Have students explain where you end up after each command.

Transition

Now for some exercises to reinforce how we work with directories.

4-8. LAB: Exercises

1. Change your current directory to `/tmp` and execute the `pwd` command. What do you see? Return to your home directory and execute a `pwd`. What is the full path name of your home directory?

2. Under your login directory there is a directory called `sample.tree`. It goes down several levels. Diagram the files and directories under `sample.tree` using the `cd`, `ls -F`, and `pwd` commands as necessary. Use ovals to indicate directory names and rectangles for regular files. Some exercises in the next module refer to files and directories somewhere under `sample.tree`.

Advanced: What option to `ls -F` could produce the names we needed using one command?

3. What is the full path name of the file `golden` under the directory `retriever` in the diagram from the previous exercise? What is its relative pathname from your home directory?

4. From your home directory, change into the `sample.tree/dog.breeds/retriever` directory. Using a relative path name, change into the `sample.tree/car.models/ford/sports` directory. Again using a relative path name, change into the `sample.tree/car.models/chevrolet` directory. Finally, return to your home directory. What commands did you use? How did you know if you arrived at each of your destinations?

5. From your home directory, obtain a directory listing of the directory `dog.breeds` under the `sample.tree` directory. Use both relative and absolute path names. What commands did you use?

4-8. LAB: Exercises

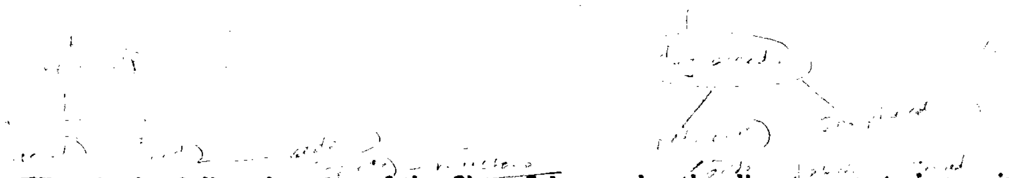
1. Change your current directory to **/tmp** and execute the **pwd** command. What do you see? Return to your home directory and execute a **pwd**. What is the full path name of your home directory?

[illegible]

2. Under your login directory there is a directory called **sample.tree**. It goes down several levels. Diagram the files and directories under **sample.tree** using the **cd**, **ls -F**, and **pwd** commands as necessary. Use ovals to indicate directory names and rectangles for regular files. Some exercises in the next module refer to files and directories somewhere under **sample.tree**.

Advanced: What option to **ls -F** could produce the names we needed using one command?

[illegible]



3. What is the full path name of the file **golden** under the directory **retriever** in the diagram from the previous exercise? What is its relative pathname from your home directory?

[illegible]

4. From your home directory, change into the **sample.tree/dog.breeds/retriever** directory. Using a relative path name, change into the **sample.tree/car.models/ford/sports** directory. Again using a relative path name, change into the **sample.tree/car.models/chevrolet** directory. Finally, return to your home directory. What commands did you use? How did you know if you arrived at each of your destinations?

[illegible]

5. From your home directory, obtain a directory listing of the directory **dog.breeds** under the **sample.tree** directory. Use both relative and absolute path names. What commands did you use?

[illegible]

6. Place yourself in your home directory. With one command, obtain a long listing of the file cp in the /bin directory. (Hint: Use the ls -l (or ll) command.) Look at the slide in topic 4-3 for a diagram of where /bin and cp are located.

Now, figure out the relative path from your home directory to the /bin directory. Obtain the same long listing of cp using a relative path name.

[illegible]

[illegible]

6. Place yourself in your home directory. With one command, obtain a long listing of the file `cp` in the `/bin` directory. (Hint: Use the `ls -l` (or `ll`) command.) Look at the slide in topic 4-3 for a diagram of where `/bin` and `cp` are located.

Now, figure out the relative path from your home directory to the `/bin` directory. Obtain the same long listing of `cp` using a relative path name.

4-8. LAB: Exercises

Instructor Notes

Purpose

Practice using the commands introduced in this module to traverse the file system.

Presentation Suggestions

Go over the exercises and draw the solutions on the board if needed. A picture of the `sample.tree` directory usually helps.

Transition

The next module introduces how to use the system's most common file manipulation commands.

Solutions

1. Change your current directory to `/tmp` and execute the `pwd` command. What do you see? Return to your home directory and execute a `pwd`. What is the full path name of your home directory?

Answer:

```
$ cd /tmp
$ pwd
/tmp
$ cd
$ pwd
/users/login_name
$
```

2. Under your login directory there is a directory called `sample.tree`. It goes down several levels. Diagram the files and directories under `sample.tree` using the `cd`, `ls -F`, and `pwd` commands as necessary. Use ovals to indicate directory names and rectangles for regular files. Some exercises in the next module refer to files and directories somewhere under `sample.tree`.

Advanced: What option to `ls -F` could produce the names we needed using one command?

Answer:

This is what `sample.tree` should look like. You could also use `ls -FR`.

```
                             sample.tree
                                (dir)
                                  |
    ----------------------------------------------------------
      |          |         |          |              |        |
  car.models   collie    honda    dog.breeds       poodle   porsche
    (dir)                           (dir)
      |                               |
  ----------------                ---------------
     |          |                  |          |
  chevrolet    ford            retriever  shepherd
   (dir)      (dir)              (dir)      (dir)
                |                  |
         -------------         -------------------------
          |          |          |            |         |
        sedan      sports     golden     labrador    mixed
        (dir)      (dir)
                     |
                  mustang
```

3. What is the full path name of the file golden under the directory `retriever` in the diagram from the previous exercise? What is its relative pathname from your home directory?

Answer:

The full path name is: `/users/login_name/sample.tree/dog.breeds/retriever/golden`
The relative path name is: `sample.tree/dog.breeds/retriever/golden`

4. From your home directory, change into the `sample.tree/dog.breeds/retriever` directory. Using a relative path name, change into the `sample.tree/car.models/ford/sports` directory. Again using a relative path name, change into the `sample.tree/car.models/chevrolet` directory. Finally, return to your home directory. What commands did you use? How did you know if you arrived at each of your destinations?

Answer:

```
$ cd sample.tree/dog.breeds/retriever
$ cd ../../car.models/ford/sports
$ cd ../../chevrolet
$ cd
```

To verify each destination:

```
$ pwd
```

5. From your home directory, obtain a directory listing of the directory `dog.breeds` under the `sample.tree` directory. Use both relative and absolute path names. What commands did you use?

Answer:

```
$ ls sample.tree/dog.breeds
$ ls /users/login_name/sample.tree/dog.breeds
```

6. Place yourself in your home directory. With one command, obtain a long listing of the file cp in the /bin directory. (Hint: Use the `ls -l` (or `ll`) command.) Look at the slide in topic 4-3 for a diagram of where /bin and cp are located.

Now, figure out the relative path from your home directory to the /bin directory. Obtain the same long listing of cp using a relative path name.

Answer:

```
$ cd
$ ls -l /bin/cp
-r-xr-xr-x   1 bin      bin         27232 Feb 17 14:00
/bin/cp

<< Relative path from a standard home directory (/users/logname) is ../../bin >>
$ ll ../../bin/cp
-r-xr-xr-x   1 bin      bin         27232 Feb 17 14:00
../../bin/cp
$
```

Module 5 — Managing Files

Objectives

Upon completion of this module, you will be able to do the following:

- List the names of currently existing files.
- Examine the contents of a file.
- Copy an existing file to a file of another name.
- Rename existing files.
- Remove existing files.

Module 5 — Managing Files

Overview of Module 5

This is the initial look at file system-type commands. The commands covered include

- `ls`—display the contents of a directory or directories.
- `cat`—display the contents of a regular ASCII file or files.
- `cp`—copy a file or files.
- `mv`—move (rename) a file or files.
- `more`—display files to the terminal screen.
- `rm`—remove a file or files.

Motivation

Most file system-related activity will necessitate using one or more of these commands. It is therefore important to learn about these basic commands before you can accomplish anything else.

The six commands covered in this module are the basis of most UNIX file system activity.

Exercises

There are short exercises dispersed throughout this module.

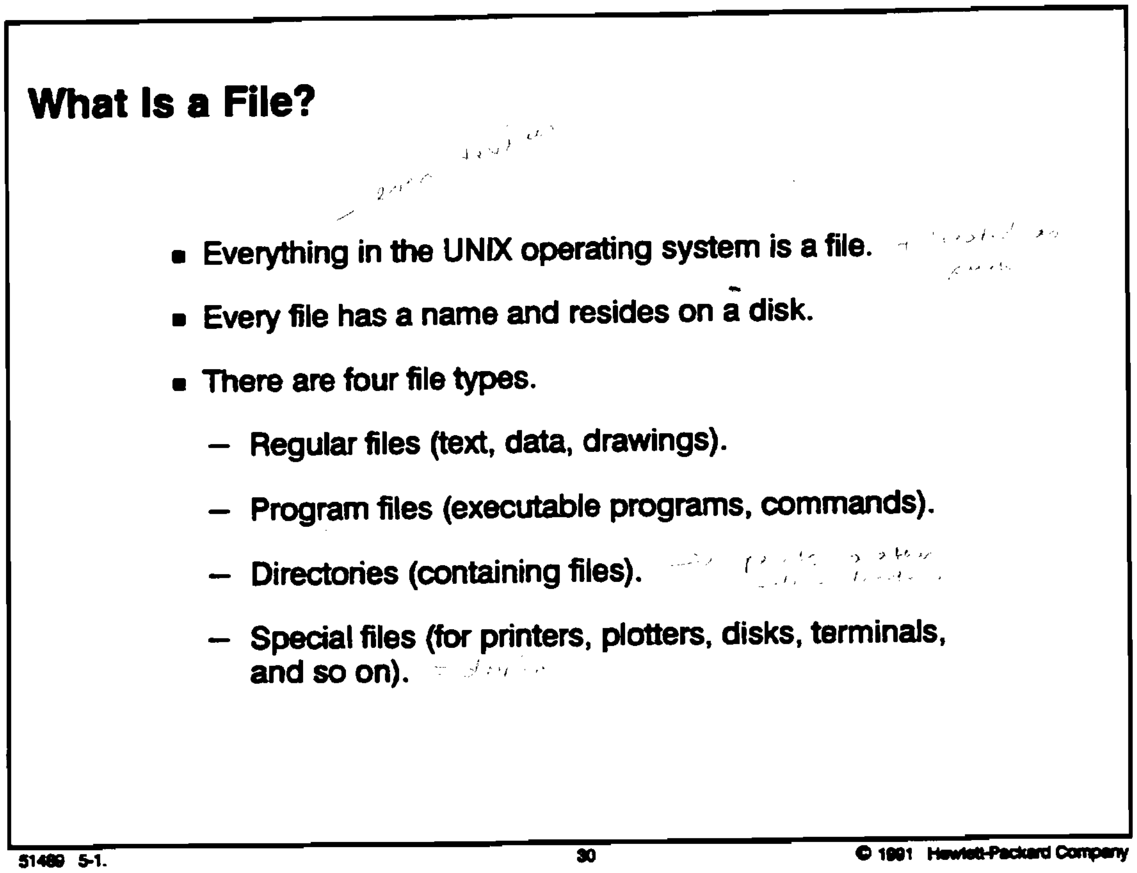
Exercise Files

`cbt`, `display1`, `display2`, `murphy`, and `reminder`

Module Transition

Now let's learn more about what we can do with files and directories.

5-1. SLIDE: What Is a File?



What Is a File?

- Everything in the UNIX operating system is a file.
- Every file has a name and resides on a disk.
- There are four file types.
  - Regular files (text, data, drawings).
  - Program files (executable programs, commands).
  - Directories (containing files).
  - Special files (for printers, plotters, disks, terminals, and so on).

51489 5-1. 30 © 1991 Hewlett-Packard Company

Student Notes

Every file has a name and takes up space. Basic types of files are:

- Data—text, drawings.
- Programs—`who`, `date`, `man`, `ls`.
- Devices—your terminal, printers, and plotters.

A **file** is a way to store information. More precisely, a UNIX file is a name and associated data stored on a mass storage device, usually a disk.

Text files are files that contain ASCII(5) characters. These files are typically created by users using a text editor at a terminal.

Program files are files that contain executable instructions. Most of the commands that we have seen are program files.

Device files are the special province of the UNIX system administrator. The UNIX operating system treats peripherals such as printers, plotters, and displays as device files.

5-1. SLIDE: What Is a File?

Purpose

HP-UX treats everything as a file. Commonly recognized files are those that contain text (strings of ASCII characters) and programs that may also be comprised of text (such as shell procedures) but more likely machine code (such as the `who` command: `/bin/who`). Even devices are treated as files.

Key Points

Explain the three categories of files.

Teaching Question

Ask if anyone can think of other device "files" (tape drives, printers, and so on).

Positive Instance

Each student's login directory is also a file. Every directory is a special type of file that contains other files and directories.

Evaluation Questions

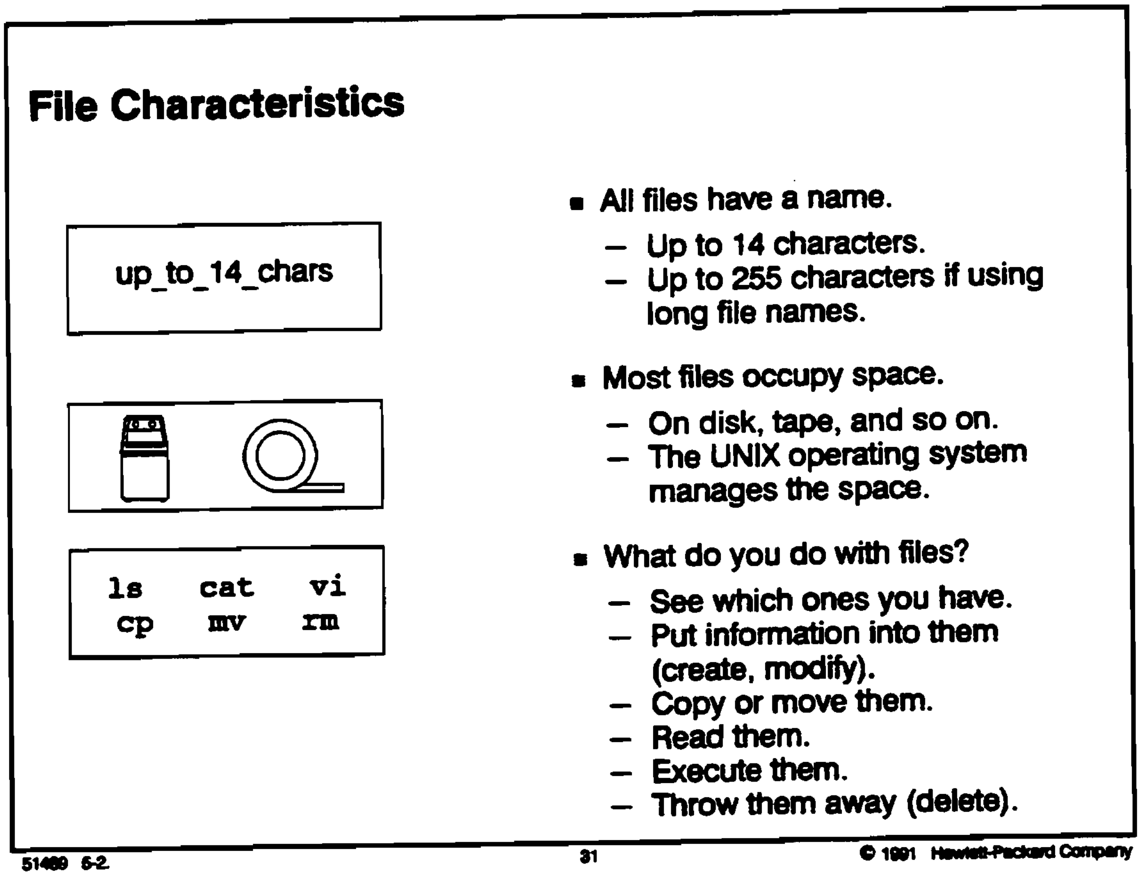
How do we know what files exist in the file system? We know by their names, using the `ls` command.

Where does HP-UX maintain most of its files? The files are maintained on the disk in the file system, which we discussed previously.

Transition

What are some of the characteristics of UNIX files?

5-2. SLIDE: File Characteristics



File Characteristics

up_to_14_chars

```
ls   cat   vi
cp   mv    rm
```

- All files have a name.
  - Up to 14 characters.
  - Up to 255 characters if using long file names.
- Most files occupy space.
  - On disk, tape, and so on.
  - The UNIX operating system manages the space.
- What do you do with files?
  - See which ones you have.
  - Put information into them (create, modify).
  - Copy or move them.
  - Read them.
  - Execute them.
  - Throw them away (delete).

51469 5-2 31 © 1991 Hewlett-Packard Company

Student Notes

A file must have a name. The name may be no more than 14 characters long. If your system administrator has enabled long file names, however, you can create file names with up to 255 characters. These names may contain any character except the ASCII null character and the "/" character. For the sake of simplicity, it is best to use just uppercase and lowercase letters (a–z and A–Z), digits (0–9), "." (dot), "_" (underscore), "-" (dash), and "+" (plus).

Associated with the name is the file's contents (data, programs, or device information). The UNIX operating system manages the space; you do not know where the data is physically stored. You do not even have to worry about finding space when you save a file or recovering that space when you delete it.

The UNIX operating system imposes no format on the contents of files. It simply sees files as a stream of characters. Some programs will expect files to be in a particular format, but that was a decision of the person who wrote the program, not a requirement of the operating system.

Most of the activity in a UNIX installation centers on files. There are many commands that manipulate files. This module introduces the most important ones.

5-2. SLIDE: File Characteristics

Purpose

A file must have a name and will usually occupy physical space on disk (for text and program files). Because most of the work accomplished on an HP-UX system involves dealing with files, we need to see the more common commands that are used to affect them.

Key Points

For systems prior to Series 300 HP-UX version 6.2 or Series 800 HP-UX version 2.0, the maximum number of characters in a file name is 14. Currently, administrators may enable long file names on a file system. This allows file names to be up to 255 characters long (see newfs(1m)).

Although the ASCII null and "/" characters cannot be used in the file name, it is normally best to avoid using all "special" characters except ".", "_", "+", and "-".

ME-10 customers need to avoid using the tilde "~" character in file names because it is used by the ME-10 application. Other applications may have the same type of restrictions.

Teaching Question

Ask students whether lowercase and uppercase letters are "seen" as the same characters to the UNIX operating system. The UNIX system is always case sensitive.

Positive Instance

The user does not have to predetermine the amount of disk space required for a file. The UNIX operating system handles all space allocation (and de-allocation) for the user.

Where Problems Arise

Invariably when students begin to create files in their login directories, some will put at least one hidden character in the file name (for example, a backspace). Other special characters such as "*", "&", and "$" may also create problems because they are shell (sh(1)) special characters.

Presentation Suggestions

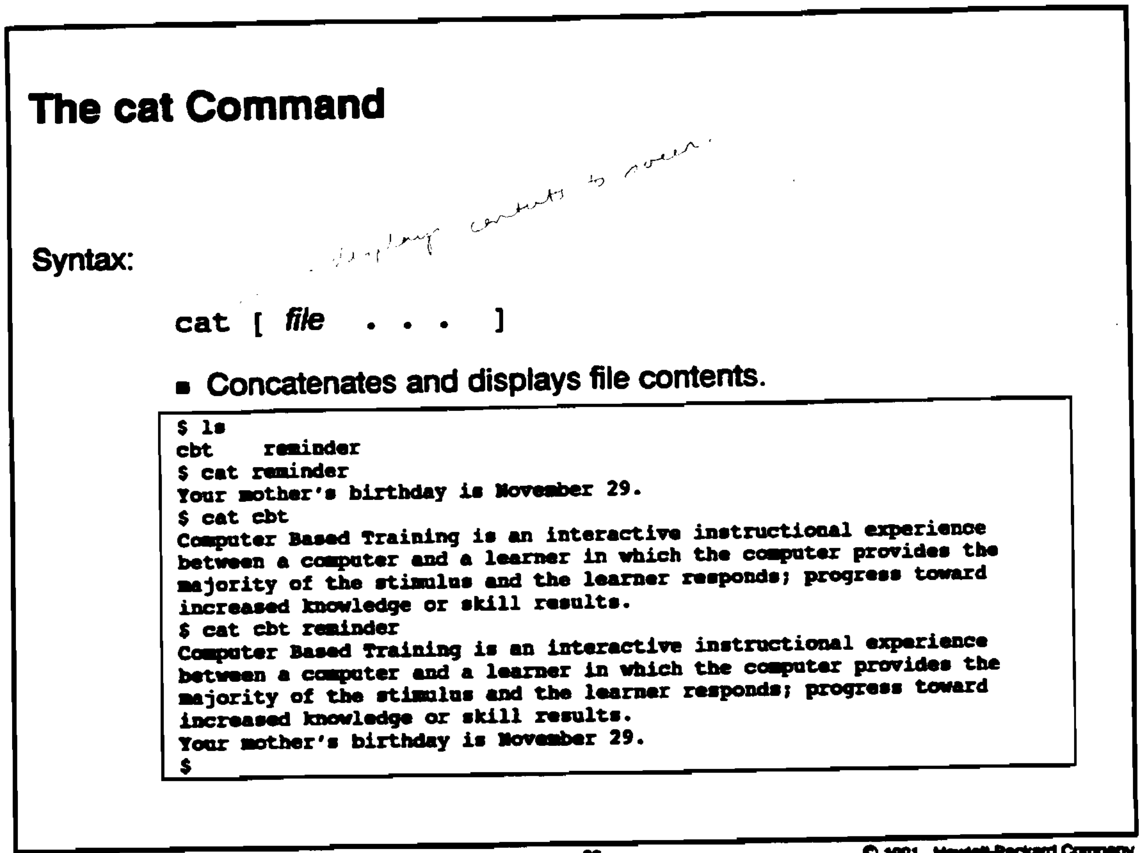
Have the students look at the ASCII chart in Section 5 of the manual. Make certain they understand that lowercase and uppercase letters are not the same.

You may also want to write the command names beside each item on the slide or on the chalkboard.

Transition

Now that we know where most files are and what kinds of things we do to them, let's take a look at these basic commands.

5-3. SLIDE: The cat Command



The cat Command

display contents to screen.

Syntax:

cat [*file* . . .]

- Concatenates and displays file contents.

```
$ ls
cbt    reminder
$ cat reminder
Your mother's birthday is November 29.
$ cat cbt
Computer Based Training is an interactive instructional experience
between a computer and a learner in which the computer provides the
majority of the stimulus and the learner responds; progress toward
increased knowledge or skill results.
$ cat cbt reminder
Computer Based Training is an interactive instructional experience
between a computer and a learner in which the computer provides the
majority of the stimulus and the learner responds; progress toward
increased knowledge or skill results.
Your mother's birthday is November 29.
$
```

51489 5-3. 32 © 1991 Hewlett-Packard Company

Student Notes

The cat command is used for concatenating and displaying text files. That is, the cat command displays the contents of a file or files. It adds no format or adornment. cat will actually display *any* type of file. If it is not a text file, however, strange things will happen to your terminal.

On the slide, the notation:

```
$ cat [ file ... ]
```

means that the cat command does not require arguments, but you may give one or more file names as arguments. We will usually use cat with just one file name argument. If you do not give cat any arguments it will read what you type at the keyboard and re-display it on your screen until you press Ctrl-d at the beginning of a line. cat has many more useful purposes than we can cover here. See cat(1) for more information.

This slide shows one use of the `cat` command, displaying files at the terminal. We `cat` the file named `reminder` and we see its contents. Then we `cat` the file named `cbt` and its contents are displayed. When we `cat` them both, they are concatenated and then displayed.

Note that we have not yet seen a method for creating files. For now we are assuming that they exist and were created at some previous time.

Exercise

1. Display the file `murphy` on your terminal using the `cat` command. What did you notice?

5-3. SLIDE: The cat Command

Purpose

The cat command "concatenates" the contents of text files to our screen so we can see the contents.

Key Points

cat will display file contents. If more than one file is given on the command line, each file is "processed" from left to right. That is, the contents of the first file are displayed followed by the contents of the next file, and so on.

Teaching Question

What will cat with an argument show you?

Answer: It will display the contents of the file.

Does it care about the size of our display?

Answer: No

Positive Instance

cat will print the contents of all named text files.

Negative Instance

cat will also display the contents of binary files and directories; however, as many of your students will no doubt discover, binary files and directory contents are *messy* when printed. Often they will contain special characters such as Ctrl-g which will make the terminal beep. They may also set some of the special modes on the terminal. You can use the tset(1) command to reset the terminal if this happens.

Where Problems Arise

It may not be clear yet to the students how this command could work if it had no arguments if the reading from stdin has not been covered.

Evaluation Questions

What happens if we `cat` a long file?

Ask for someone to explain which output line comes from `reminder` and which comes from `cbt` in the slide example.

Slide Notes

Use the "reveal" method, distinctly showing the command line first.

Transition

Let's look at a way to examine the contents of files one screen at a time.

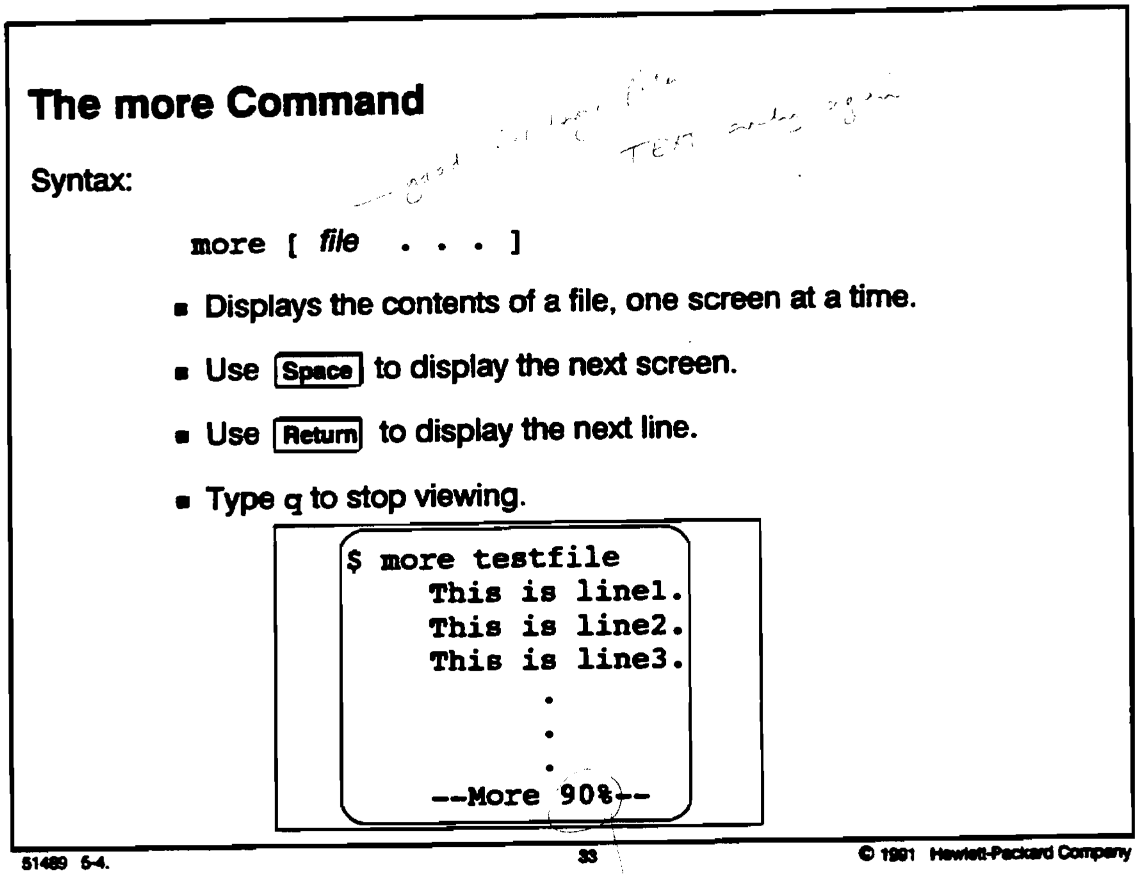
Answer

1. Display the file `murphy` on your terminal using the `cat` command. What did you notice?

Answer:

The `cat` command simply concatenaters the file to your screen without regard to the size of the screen. It scrolls the text by until it reaches the end of the file

5-4. SLIDE: The more Command



The more Command

[illegible]

Syntax:

```
more [ file . . . ]
```

- Displays the contents of a file, one screen at a time.
- Use Space to display the next screen.
- Use Return to display the next line.
- Type q to stop viewing.

```
$ more testfile
    This is line1.
    This is line2.
    This is line3.
              .
              .
              .
    --More 90%--
```

51489 5-4. 33 © 1991 Hewlett-Packard Company

% displayed

Student Notes

The more command displays the contents of the named files. Unlike the cat command, it knows how many lines are on your terminal and it will only print one screen of material at a time. If the file is more than one screen long, more will display a prompt at the bottom of the screen indicating how much of the material it has displayed.

To see the next screen of material, press the space bar. If you want to see just the next line of material, you can press Return. If you want to stop viewing the file, press q.

When you are displaying files to your screen, you will find the more command very useful.

The format is:

```
$ more [ file ... ]
```

Exercises

1. You have already used the `cat` command to display `murphy` on your screen. Now try to display `murphy` with the `more` command.

2. Use the `more` command to display both `cbt` and `reminder` at once.

5-4. SLIDE: The more Command

Instructor Notes

Purpose

The `cat` command can be difficult to use on a terminal when the user is viewing large files. The `more` command displays files intelligently.

Key Points

The `more` command allows you to intelligently view a file at a terminal screen because it knows how many lines are on the terminal.

Teaching Question

How would you examine a large file at a screen if you did not have the `more` command? The only way is to use Ctrl-s to stop the display from scrolling off the screen and Ctrl-q to restart the display. Note that not all vendors' terminals can do this.

Where Problems Arise

This is the first command that is interactive. Students may become confused because it expects a different type of response.

Transition

We can look at what else can we do with files.

Answers

1. You have already used the `cat` command to display `murphy` on your screen. Now try to display `murphy` with the `more` command.

Answer:

```
$ more murphy
```

Notice that the file is displayed one screen at a time. The space bar displays the next screen and Return displays just the next line.

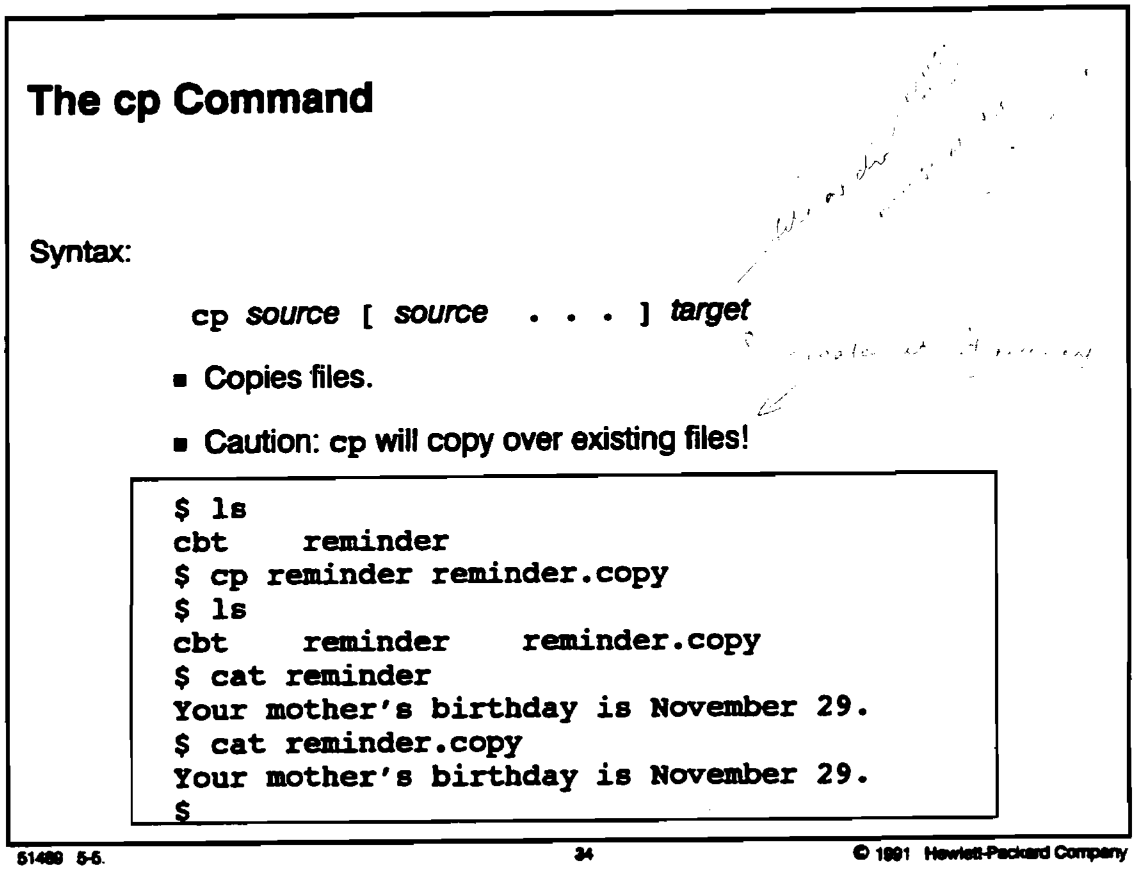
2. Use the `more` command to display both `cbt` and `reminder` at once.

Answer:

```
$ more cbt reminder
:::::::::::::
cbt
:::::::::::::
Computer Based Training is an interactive instructional
experience between a computer and a learner in which the
computer provides the increased knowledge or skill results.
:::::::::::::
reminder
:::::::::::::
Your mother's birthday is November 29.
$
```

Notice that `more` prompted you to start displaying the second file after it displayed the first file.

5-5. SLIDE: The cp Command



The cp Command

Syntax:

cp *source* [*source* . . .] *target*

- Copies files.
- Caution: cp will copy over existing files!

```
$ ls
cbt    reminder
$ cp reminder reminder.copy
$ ls
cbt    reminder    reminder.copy
$ cat reminder
Your mother's birthday is November 29.
$ cat reminder.copy
Your mother's birthday is November 29.
$
```

51489 5-5. 34 © 1991 Hewlett-Packard Company

Student Notes

The cp command is used to make a copy of a file. The general command:

$ cp *source* [*source* ...] *target*

copies the contents of the file named by *source* into a file named *target*. If *target* did not previously exist, the cp command creates it. If *target* did previously exist, its contents are *replaced* by the contents of the file named *source*!

This is typical of the UNIX operating system. The system will not check your work. It assumes that you know what you are doing. If your desire is to overwrite *target*, it will not argue with you.

The last argument may be a file name or a directory. If it is a directory, the file named by *source* will be copied into a file named *source* in the directory named *target* (which must already exist). If you specify more than one source file, the target *must* be a directory.

Here are a few examples of cp:

- cp will write over existing files:

```
$ ls
f1  f2
$ cp f1 f2
$ ls
f1  f2
$ cat f1
This is a line of text in f1.
$ cat f2
This is a line of text in f1.
$
```

The file f2 was overwritten!

- cp will not copy a file to itself:

```
$ cp f1 f1
cp: f1 and f1 are identical
$
```

- cp can copy several files into another directory:

```
$ cp f1 f2 junkdir
$ ls
f1   f2   junkdir
$ ls junkdir
f1   f2
$
```

5-5. SLIDE: The cp Command

Purpose

cp is a way of copying files. cp creates an exact copy of the contents of one file on disk and gives the copy a different name.

Key Points

cp will overwrite a file that exists if you specify that name as the target.

If more than one source file is given, then the last argument (*target*) must be a directory.

Where Problems Arise

Students usually do not like the idea that cp will let them overwrite existing files without even prompting them. This is the philosophy of the UNIX operating system that we covered in Module 3. No doubt, you will some day be "burned" by this "feature."

Evaluation Questions

Ask how many arguments cp can have (unless you have already dealt with this question). Of course, the "number" of arguments is not the limiting factor. Each command line normally has a limit of 1024 characters, which should be more than enough.

Presentation Suggestions

Just show the slide and explain it. Make sure there are no questions before continuing.

Slide Notes

Use the "reveal" method, showing the command line followed by output.

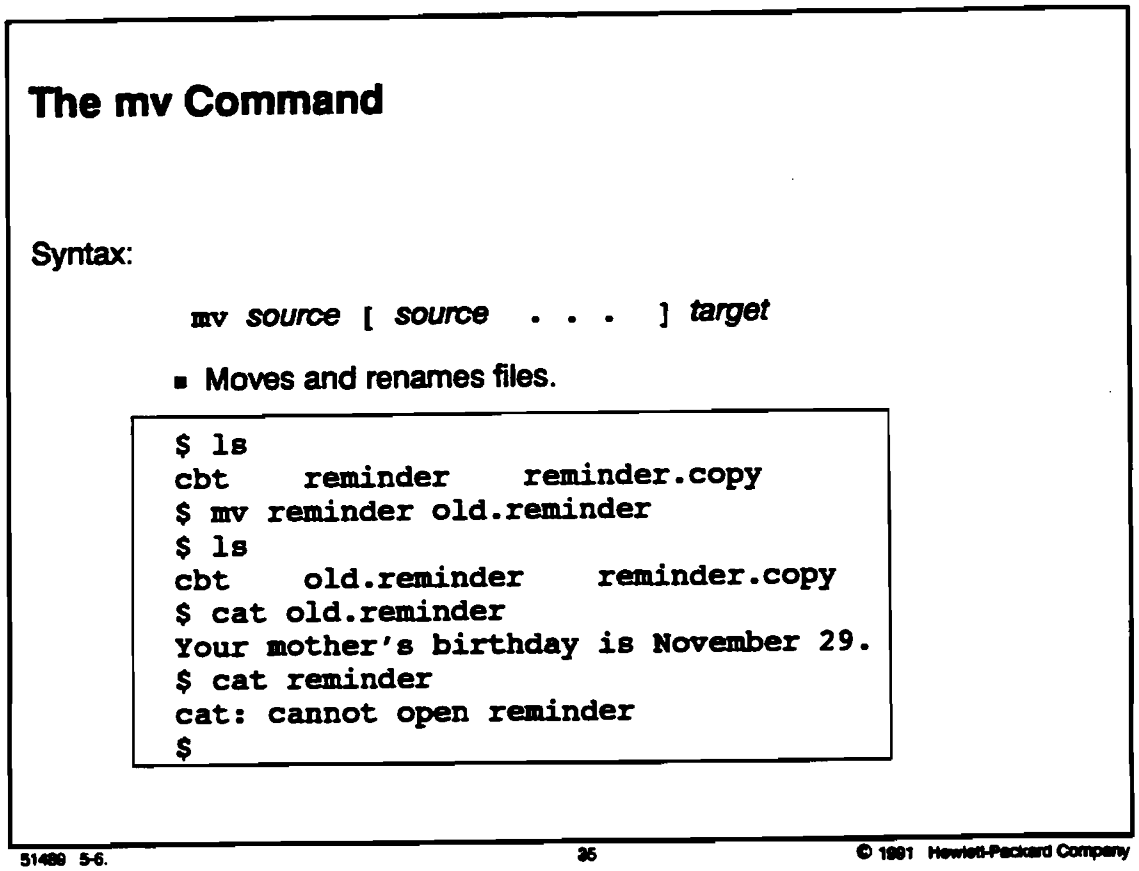
Chalkboard Notes

You may want to review the examples given in the text on the board.

Transition

Now we will look at a command that "renames" files.

5-6. SLIDE: The mv Command



The mv Command

Syntax:

`mv` *source* [*source* . . .] *target*

- Moves and renames files.

```
$ ls
cbt    reminder    reminder.copy
$ mv reminder old.reminder
$ ls
cbt    old.reminder    reminder.copy
$ cat old.reminder
Your mother's birthday is November 29.
$ cat reminder
cat: cannot open reminder
$
```

51489 5-6. 35 © 1991 Hewlett-Packard Company

Student Notes

The `mv` command is used to rename files. Its format is very similar to the `cp` command:

`$ mv` *source* [*source* ...] *target*

This will rename the file named by *source* to *target*. If a file named `target` previously existed, its contents are replaced silently (as you would expect by now).

If *target* is a directory, then the named files will be moved to the directory named *target*, keeping their original names:

```
$ ls
f1  f2  junkdir
$ mv f1 f2 junkdir
$ ls
junkdir
$ ls junkdir
f1  f2
$
```

About the slide: The `ls` shows that we are starting with three files. After the first `mv`, the second `ls` shows that we still have only three files, but the name of `reminder` has been changed to `old.reminder`. The final `cat` shows what happens if one tries to `cat` a nonexistent file.

5-6. SLIDE: The mv Command

Instructor Notes

Purpose

mv moves and renames files while cp creates a duplicate with a different name.

Key Points

Similar to cp, the mv command creates a new file name. The old file name, however, is gone.

Teaching Question

Can a file be moved to itself?

Answer: No!

Where Problems Arise

Some students may still have a tendency to enter "hidden" characters in the *target* file names. The classic combination is typing "\" Backspace. This actually leaves a backspace on the command line!

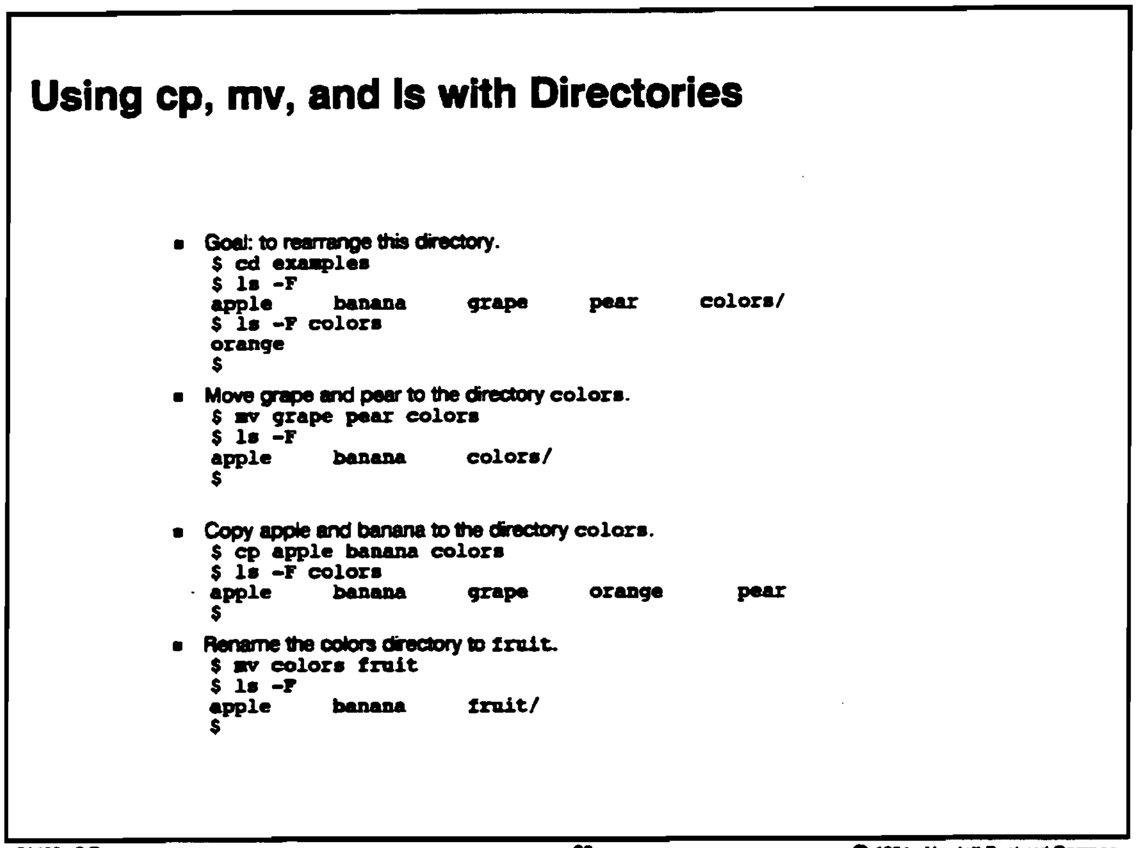
Presentation Suggestions

Discuss the directory example given in the text.

Transition

We can use these commands between directories.

5-7. SLIDE: Using cp, mv, and ls with Directories



Using cp, mv, and ls with Directories

- Goal: to rearrange this directory.

```
$ cd examples
$ ls -F
apple     banana     grape     pear     colors/
$ ls -F colors
orange
$
```

- Move grape and pear to the directory colors.

```
$ mv grape pear colors
$ ls -F
apple     banana     colors/
$
```

- Copy apple and banana to the directory colors.

```
$ cp apple banana colors
$ ls -F colors
apple     banana     grape     orange     pear
$
```

- Rename the colors directory to fruit.

```
$ mv colors fruit
$ ls -F
apple     banana     fruit/
$
```

51489 5-7. 36 © 1991 Hewlett-Packard Company

Student Notes

We have seen that both cp and mv can be used with a directory as the target of the operation.

The mv command has one capability with directories that the cp command does not have. The mv command allows you to change the name of a directory:

```
$ mv  old_dir_name  new_dir_name
```

Take a look at mv(1) in the manual for more detail.

The following are the basic forms of the cp command:

Copy a file	`$ cp file1 file2`
Copy a file to a directory	`$ cp file1 dir1`
Copy several files to a directory	`$ cp file1 file2 ... filen dir1`

The following are the basic forms of the mv command:

Rename a file	`$ mv file1 file2`
Move a file to another directory	`$ mv file1 dir1`
Move several files to another directory	`$ mv file1 file2 ... filen dir1`
Rename a directory	`$ mv dir1 dir2`

5-7. SLIDE: Using cp, mv, and ls with Directories

Instructor Notes

Purpose

We can move or copy many files at once.

Key Points

The `mv` and `cp` commands work with directories as the target.

Slide Notes

Use "reveal" to show what is going on, one step at a time. Take time to answer questions and point out each command.

Teaching Question

How can we copy the `fruit` directory to a new directory called `my_fruit` in one command? (Hint: There is an option to cp that can do this.)

```
$ cp -r fruit my_fruit
```

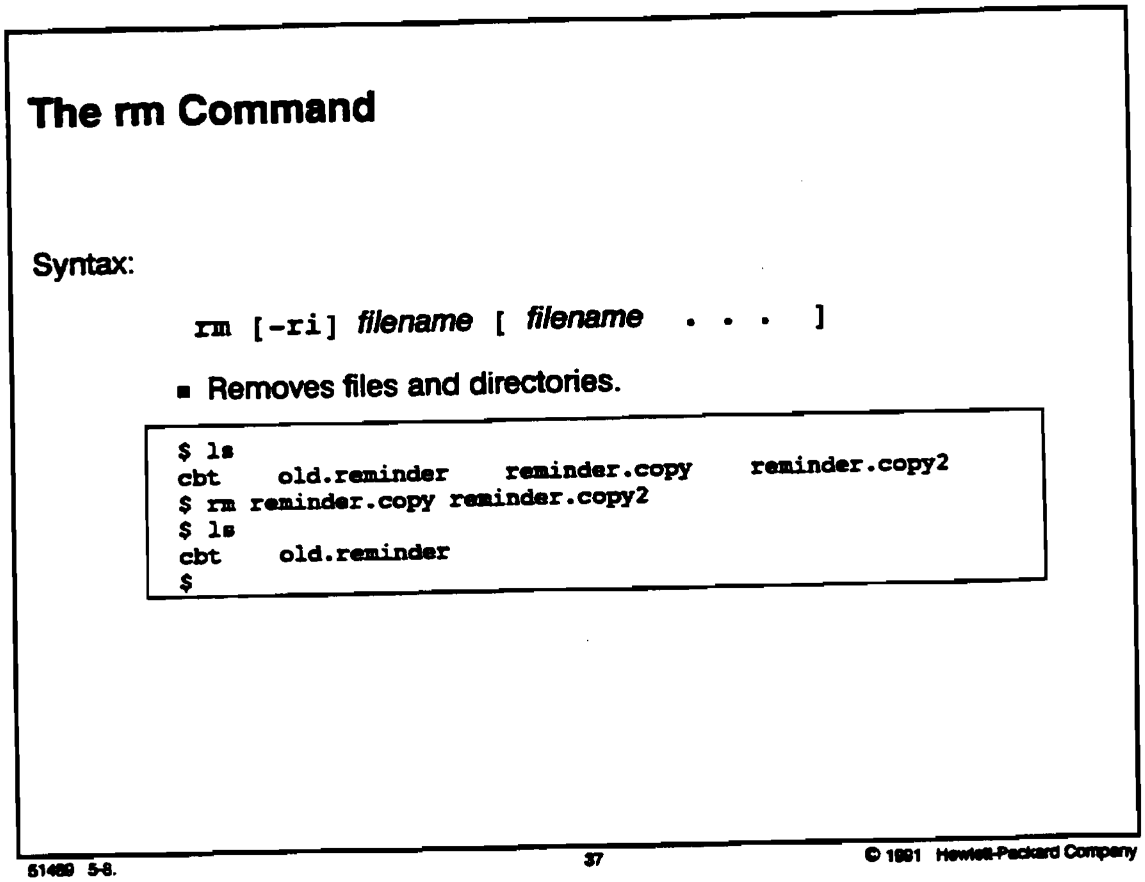
Chalkboard Notes

If students need help, you can draw the directory structure on the board as you go through the slide.

Transition

Now that we have seen how to create files, let's look at the command used to delete files.

5-8. SLIDE: The rm Command



The rm Command

Syntax:

```
rm [-ri] filename [ filename . . . ]
```

- Removes files and directories.

```
$ ls
cbt    old.reminder    reminder.copy    reminder.copy2
$ rm reminder.copy reminder.copy2
$ ls
cbt    old.reminder
$
```

51489 5-8. 37 © 1991 Hewlett-Packard Company

Student Notes

The rm command is used to remove files. Once removed, they *cannot* be retrieved unless the system administrator has created backup copies. The rm command must have at least one file name as an argument and will accept many. If more than a single argument is given, all arguments will be removed by rm.

The slide shows the most commonly used options:

-r	Recursively deletes the entire contents of the specified directory and the directory itself.
-i	Interactive, rm asks whether or not to delete each file.

The -r option "recursively" removes the contents of any directories named on the command line. For example:

```
$ ls
f1  f2  junkdir
$ ls junkdir
file3  file4
$ rm -r junkdir
$ ls
f1 f2
$
```

We caution against indiscriminate use of the `-r` option.

On the other hand we do recommend the use of the `-i` option. If you use the `-i`, meaning interactive, the `rm` command prints each file name and asks you whether or not you really want to delete it. If you respond with a `y`, the file is removed; otherwise it is left intact.

These options are discussed further in rm(1).

5-8. SLIDE: The rm Command — Instructor Notes

Purpose

To explain the format and syntax of how to remove one or more files or directories. By using the `ls` and `cat` examples, we prove that the files have been deleted.

Key Points

The `rm` command requires at least one "valid" file name.

The `-r` option will also remove named directories and their subdirectories.

Teaching Question

Ask how `rm` notifies the user if it deletes the named file or files.

Answer: It does not notify the user! `rm`, the same as most UNIX system commands, works silently.

Positive Instance

All specified files can be deleted by just one invocation of the `rm` command.

Negative Instance

All of your files can be deleted by just one invocation of the `rm` command!

Be certain that a deletion is what is wanted because once removed, it is gone!

Where Problems Arise

One or more files may be inadvertently removed. This is where the shell special characters can really cause problems, especially "`*`" because it is shorthand for "all file names." This is why it is good to keep a copy of the lab files online!

Chalkboard Notes

Give examples of the `-r` and `-i` options on the board if needed.

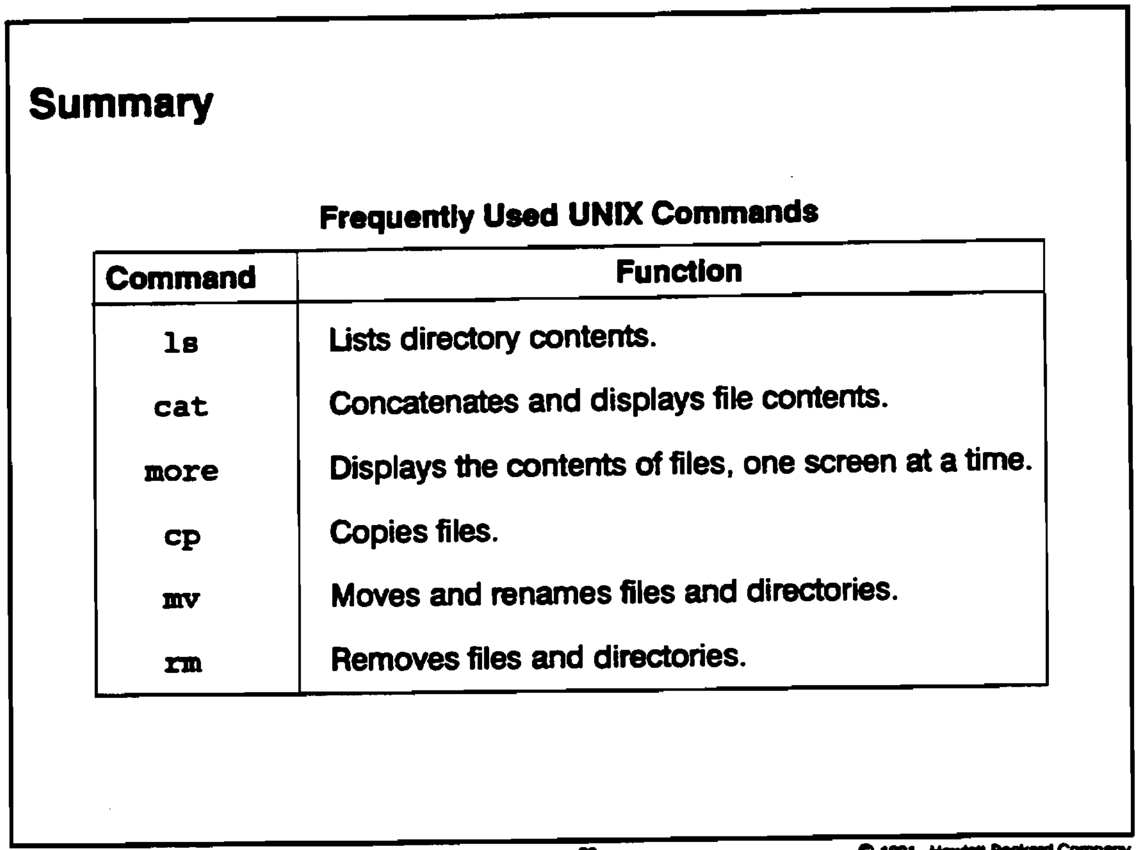
Slide Notes

Using the "reveal" method, point out each command line and explain it.

Transition

Here is a summary of the "file" commands reviewed so far.

5-9. SLIDE: Summary



Summary

Frequently Used UNIX Commands

Command	Function
`ls`	Lists directory contents.
`cat`	Concatenates and displays file contents.
`more`	Displays the contents of files, one screen at a time.
`cp`	Copies files.
`mv`	Moves and renames files and directories.
`rm`	Removes files and directories.

51489 5-9. 38 © 1991 Hewlett-Packard Company

Student Notes

These are six of the most frequently used commands in a UNIX system. To be a successful user, you must be familiar with their function and syntax.

The three commands `ls`, `cat`, and `more` operate with the two different aspects of UNIX files: `ls` deals with file names, while `cat`, and `more` deal with contents.

The two commands `cp` and `mv` cause the data of a file to be known by some other name.

At the conclusion of a `cp` command, two identical files exist, one with the name specified by the first argument on the command line and one with the name specified by the second argument. At the conclusion of an `mv` command, only one file exists, with the name specified by the command line's second argument. For a summary of both commands, see cp(1) and mv(1).

The `rm` command is used to remove files and directories.

5-9. SLIDE: Summary

Purpose

This is just to reinforce what has been covered so far using file system commands.

Key Points

These are the most frequently used file system commands. Be certain students understand their use.

Evaluation Question

Ask the difference between `cp` and `mv`.

Presentation Suggestions

Review the commands as shown on the slide.

Transition

Next are the terminal exercises for this module.

5-10. LAB: Exercises

Use the *HP-UX Reference Manual* to look up details about commands and their usage.

1. Notice the two files named `display1` and `display2`. See what each contains by displaying their contents on your terminal. Try to accomplish this using a single command line.

2. Make a copy of `display1` and call it `More_Than_14_Chars`. Now do an `ls`. What happens?

3. Make a copy of `display1` and call it `display1.copy`. Make a copy of `display2` and call it `display2.copy`.

4. Use the `ls` command to verify that both `display1.copy` and `display2.copy` exist. Do this so that the `ls` command outputs only the names of the files `display1.copy` and `display2.copy` and not the names of any other files.

5. Try to make a copy of `display1.copy` and call it `display2.copy`. Display the contents of `display2.copy`. What happened?

6. Rename `display2.copy` as `display1.copy1`.

7. Now remove all of the files that you have been responsible for creating. Use a single command line.

8. Change your current directory to `sample.tree`. In one command, copy `collie` and `poodle` into `dog.breeds`. Perform this from whatever directory is most convenient. You might need to look up the `cp` command. (Hint: `dog.breeds` will be the *target* of the copy.)

9. In one command, move `honda` and `porsche` into `car.models`. Perform this from whatever directory is most convenient. You might need to look up the `mv` command.

10. In one `ls` command, verify that the `cp` and `mv` worked correctly.

11. Rename the `sample.tree` directory to `cars+dogs`.

12. Use the `rm` command with the `-i` option to remove the file `rm_me`.

Advanced Exercises

1. The `ln` command is used to attach more than one name to the same file space. If two files are linked, you can change one and both change. This is handy for sharing common files such as customization files. `ln` uses the same syntax as the `cp` and `mv` commands.

In your home directory there is a file called **`reminder`**. Link it to another name such as **`reminder.link`**. `cat` both files to prove that they are identical.

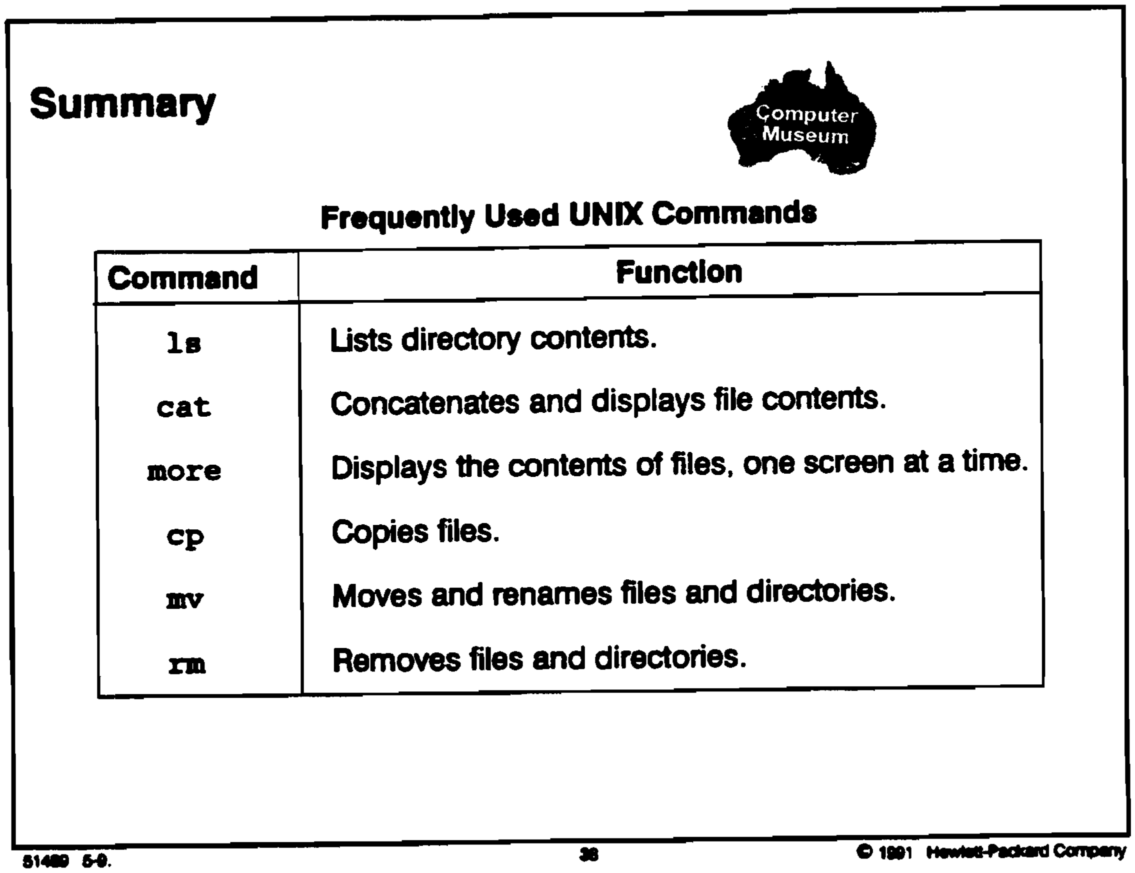
Change **`reminder`** with the command:

```
echo "Meeting 6/21" >> reminder
```

(We will see how this works in Module 9.) `cat` both files again. What do you see?

2. What happens if you remove **`reminder.link`**?

5-9. SLIDE: Summary



Summary

Frequently Used UNIX Commands

Command	Function
`ls`	Lists directory contents.
`cat`	Concatenates and displays file contents.
`more`	Displays the contents of files, one screen at a time.
`cp`	Copies files.
`mv`	Moves and renames files and directories.
`rm`	Removes files and directories.

51489 5-9. 38 © 1991 Hewlett-Packard Company

Student Notes

These are six of the most frequently used commands in a UNIX system. To be a successful user, you must be familiar with their function and syntax.

The three commands `ls`, `cat`, and `more` operate with the two different aspects of UNIX files: `ls` deals with file names, while `cat`, and `more` deal with contents.

The two commands cp and `mv` cause the data of a file to be known by some other name.

At the conclusion of a cp command, two identical files exist, one with the name specified by the first argument on the command line and one with the name specified by the second argument. At the conclusion of an `mv` command, only one file exists, with the name specified by the command line's second argument. For a summary of both commands, see cp(1) and mv(1).

The `rm` command is used to remove files and directories.

5-10. LAB: Exercises

Use the *HP-UX Reference Manual* to look up details about commands and their usage.

1. Notice the two files named **display1** and **display2**. See what each contains by displaying their contents on your terminal. Try to accomplish this using a single command line.

   cat display1 display2

2. Make a copy of **display1** and call it **More_Than_14_Chars**. Now do an **ls**. What happens?

   [illegible]

3. Make a copy of **display1** and call it **display1.copy**. Make a copy of **display2** and call it **display2.copy**.

   cp display1 display1.copy

4. Use the **ls** command to verify that both **display1.copy** and **display2.copy** exist. Do this so that the **ls** command outputs only the names of the files **display1.copy** and **display2.copy** and not the names of any other files.

   ls display

5. Try to make a copy of **display1.copy** and call it **display2.copy**. Display the contents of **display2.copy**. What happened?

   [illegible]

6. Rename **display2.copy** as **display1.copy1**.

   [illegible]

7. Now remove all of the files that you have been responsible for creating. Use a single command line.

rm display.c copy here [illegible] _ ll _ char [illegible]

8. Change your current directory to **sample.tree**. In one command, copy **collie** and **poodle** into **dog.breeds**. Perform this from whatever directory is most convenient. You might need to look up the **cp** command. (Hint: **dog.breeds** will be the *target* of the copy.)

cd [illegible]

[illegible] collie poodle dog.breeds

9. In one command, move **honda** and **porsche** into **car.models**. Perform this from whatever directory is most convenient. You might need to look up the **mv** command.

[illegible]

mv honda porsche car.models

10. In one **ls** command, verify that the **cp** and **mv** worked correctly.

[illegible]

11. Rename the **sample.tree** directory to **cars+dogs**.

[illegible]

12. Use the **rm** command with the **-i** option to remove the file **rm_me**.

[illegible]

hard [illegible]

Advanced Exercises

1. The `ln` command is used to attach more than one name to the same file space. If two files are linked, you can change one and both change. This is handy for sharing common files such as customization files. `ln` uses the same syntax as the `cp` and `mv` commands.

In your home directory there is a file called `reminder`. Link it to another name such as `reminder.link`. `cat` both files to prove that they are identical.

Change `reminder` with the command:

```
echo "Meeting 6/21" >> reminder
```

(We will see how this works in Module 9.) `cat` both files again. What do you see?

[illegible]

2. What happens if you remove `reminder.link`?

[illegible]

[illegible]

[illegible]

5-10. LAB: Exercises

Purpose

Practice using commands to manipulate files.

Transition

Let's look at some advanced directory and file commands.

Solutions

Use the *HP-UX Reference Manual* to look up details about commands and their usage.

1. Notice the two files named display1 and display2. See what each contains by displaying their contents on your terminal. Try to accomplish this using a single command line.

Answer:

```
$ more display1 display2
::::::::::::::
display1
::::::::::::::
This is the file display1
::::::::::::::
display2
::::::::::::::
This is the file display2
```

Note that cat would have worked also without the separation between files.

2. Make a copy of display1 and call it More_Than_14_Chars. Now do an ls. What happens?

Answer:

```
$ cp display1 More_Than_14_Chars
$ ls
display1 More_Than_14_C
```

If you are on a system using long file names, the complete file name will be used.

3. Make a copy of display1 and call it display1.copy. Make a copy of display2 and call it display2.copy.

Answer:

```
$ cp display1 display1.copy
$ cp display2 display2.copy
```

4. Use the `ls` command to verify that both `display1.copy` and `display2.copy` exist. Do this so that the `ls` command outputs only the names of the files `display1.copy` and `display2.copy` and not the names of any other files.

Answer:

```
$ ls display1.copy display2.copy
display1.copy    display2.copy
```

5. Try to make a copy of `display1.copy` and call it `display2.copy`. Display the contents of `display2.copy`. What happened?

Answer:

```
$ cp display1.copy display2.copy
$ cat display2.copy
This is the file display1
```

The contents of `display2.copy` were overwritten with `display1.copy`.

6. Rename `display2.copy` as `display1.copy1`.

Answer:

```
$ mv display2.copy display1.copy1
```

7. Now remove all of the files that you have been responsible for creating. Use a single command line.

Answer:

```
$ rm display1.copy1 display1.copy More_Than_14_c
```

8. Change your current directory to `sample.tree`. In one command, copy `collie` and `poodle` into `dog.breeds`. Perform this from whatever directory is most convenient. You might need to look up the `cp` command. (Hint: `dog.breeds` will be the *target* of the copy.)

Answer:

```
$ cp collie poodle dog.breeds
```

9. In one command, move `honda` and `porsche` into `car.models`. Perform this from whatever directory is most convenient. You might need to look up the `mv` command.

Answer:

```
$ mv honda porsche car.models
```

10. In one ls command, verify that the cp and mv worked correctly.

Answer:

```
$ ls . car.models dog.breeds
.:
car.models    collie    dog.breeds    poodle

car.models:
chevrolet   ford   honda   porsche

dog.breeds:
collie      poodle     retriever     shepherd
```

11. Rename the sample.tree directory to cars+dogs.

Answer:

```
$ cd
$ mv sample.tree cars+dogs
```

12. Use the rm command with the -i option to remove the file rm_me.

Answer:

```
$ rm -i rm_me
rm_me?  y
$
```

Advanced Exercises

1. The ln command is used to attach more than one name to the same file space. If two files are linked, you can change one and both change. This is handy for sharing common files such as customization files. ln uses the same syntax as the cp and mv commands.

In your home directory there is a file called reminder. Link it to another name such as reminder.link. cat both files to prove that they are identical.

Change reminder with the command:

```
echo "Meeting 6/21" >> reminder
```

(We will see how this works in Module 9.) cat both files again. What do you see?

Answer:

```
$ ln reminder reminder.link
$ cat reminder
Your mother's birthday is November 29.
$ cat reminder.link
```

```
Your mother's birthday is November 29.
$ echo "Meeting 6/21" >> reminder
$ cat reminder
Your mother's birthday is November 29.
Meeting 6/21
$ cat reminder.link
Your mother's birthday is November 29.
Meeting 6/21
```

Notice that you changed only the file `reminder` but because the two files were linked, `reminder.link` was also changed.

2. What happens if you remove `reminder.link`?

Answer:

```
$ rm reminder.link
$ ls reminder reminder.link
reminder.link not found
reminder
$
```

Notice that the file names really are referencing the same space on the disk. Removing one of the links, however, leaves the other file intact.

Module 6 — More on Files and Directories

Objectives

Upon completion of the module, you will be able to do the following:

- Explain the meaning of file permissions.
- Change file and directory permissions.
- Describe and change the owner and group attributes of a file.
- Create and remove directories.
- Use basic filename generating characters.

Module 6 — More on Files and Directories

Overview of Module 6

This module provides an introduction to some of the more complex file and directory operations. If you have an advanced class, this is a good way to get them some of the "good stuff." If your class is having trouble with the material so far, you may consider handling this module as an "optional" module.

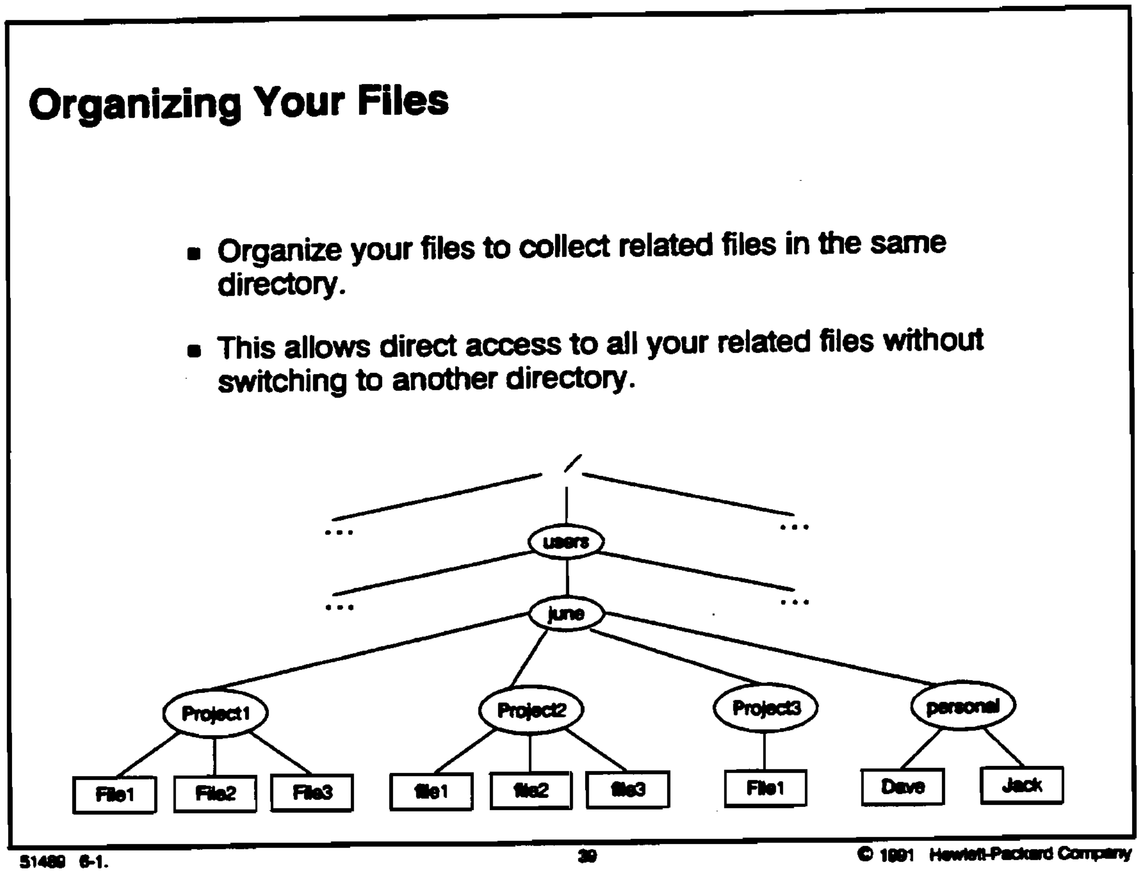
Motivation

You can "get by" with the commands that we have seen so far, but to effectively use the HP-UX system, you will need the commands and concepts covered in this module.

Module Transition

Now that we know how to manipulate files, let's take a look at how we can actually edit the contents of our text files.

6-1. SLIDE: Organizing Your Files


/
...
...
users
...
...
june
Project1
Project2
Project3
personal
File1
File2
File3
file1
file2
file3
File1
Dave
Jack

Organizing Your Files

- Organize your files to collect related files in the same directory.
- This allows direct access to all your related files without switching to another directory.

51489 6-1. 39 © 1991 Hewlett-Packard Company

Student Notes

Directories are powerful because they can contain other directories as well as files. This leads to a hierarchy of directories. You can store a file in a directory that is stored in another directory, which may be stored in another directory, and so on.

You can organize your files in a way that collects related files in the same directory. One good example is the use of directories to store files (or other directories) related to a particular project. This is similar to the way you use folders in a file cabinet to store related documents.

The slide shows an example of how to organize your files.

Overall, it is easier to manage more directories that contain fewer files than to have only a few directories containing many files. You will have to find the balance point that best suits your taste.

6-1. SLIDE: Organizing Your Files

Instructor Notes

Purpose

By placing files into certain directories, you save time searching unrelated directories for your files.

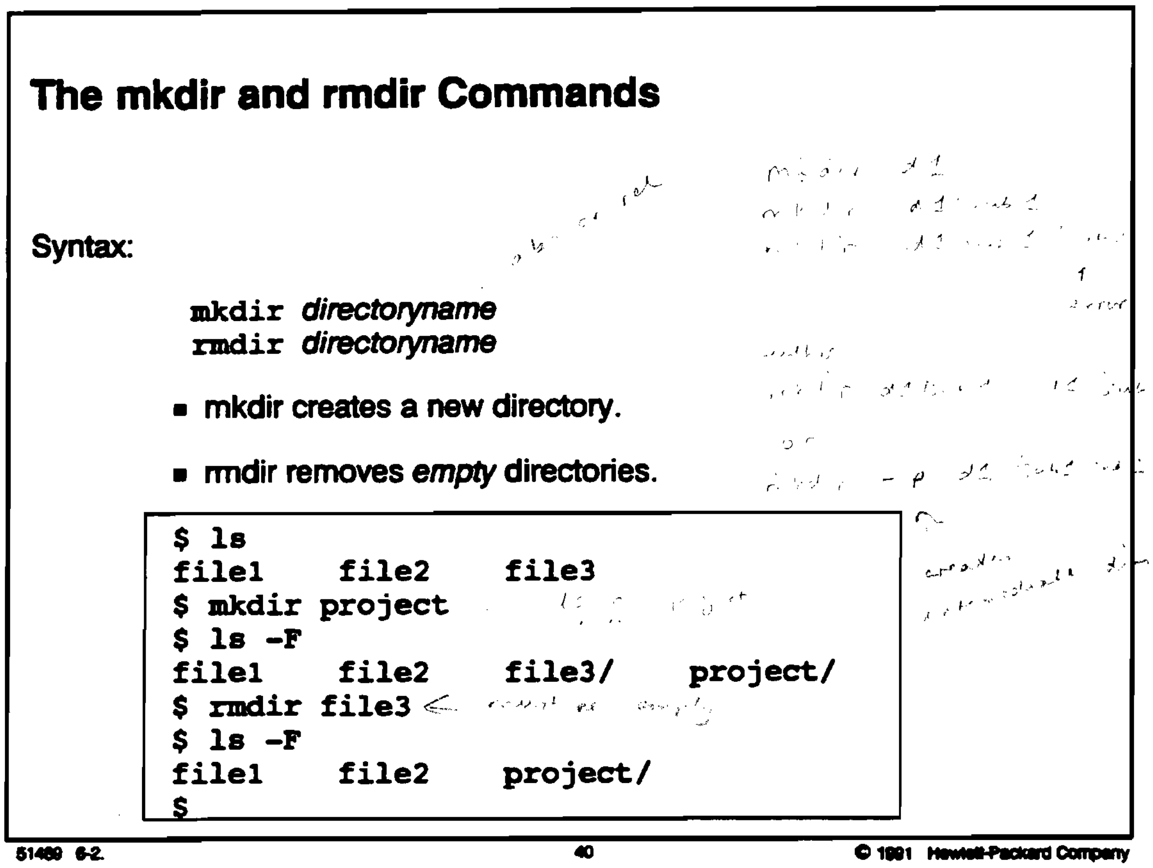
Key Points

While in the same directory, you can have direct access to all of your related files without having to switch to another directory.

Transition

If we are going to store our files in directories, we need a way to create those directories.

6-2. SLIDE: The mkdir and rmdir Commands



The mkdir and rmdir Commands

Syntax:

```
mkdir directoryname
rmdir directoryname
```

- mkdir creates a new directory.
- rmdir removes *empty* directories.

```
$ ls
file1    file2    file3
$ mkdir project
$ ls -F
file1    file2    file3/    project/
$ rmdir file3
$ ls -F
file1    file2    project/
$
```

[illegible]

51469 6-2. 40 © 1991 Hewlett-Packard Company

Student Notes

The `mkdir` command allows us to make directories. We can use these new directories to help organize our files. When each directory is created, it is automatically endowed with "." and "..". Note that creating directories does not change your current directory. What is the only way to change directories?

The `rmdir` command allows us to remove directories. A directory must be empty in order to be removed with `rmdir` (that is, it contains no entries other than "." and ".."). It may be easier to use `rm -r` if you want to remove a directory that contains files. Also, no directory between your current directory and the root may be removed.

Some people use all capital letters to name directories so that they are easily distinguished from regular files. Most people, however, use the `ls -F` command to differentiate between files and subdirectories in the current directory.

6-2. SLIDE: The mkdir and rmdir Commands

Instructor Notes

Purpose

To illustrate the commands for creating and deleting directories.

Key Points

When you create a directory with `mkdir`, the new directory contains only "." and "..". A directory can be deleted using `rmdir` only when it is empty (that is, when it has no entries other than "." and ".."). We saw how to remove directories containing files in Module 5.

Teaching Question

How can `ls -F` show which names are directories?

Answer: A slash (/) is appended to each file that is a directory.

What one command will remove a directory that contains files?

Answer: The command is `rm -r`.

Where Problems Arise

Creating directories in the wrong subdirectory causes problems. Students may need to be reminded of `cd` and `pwd`.

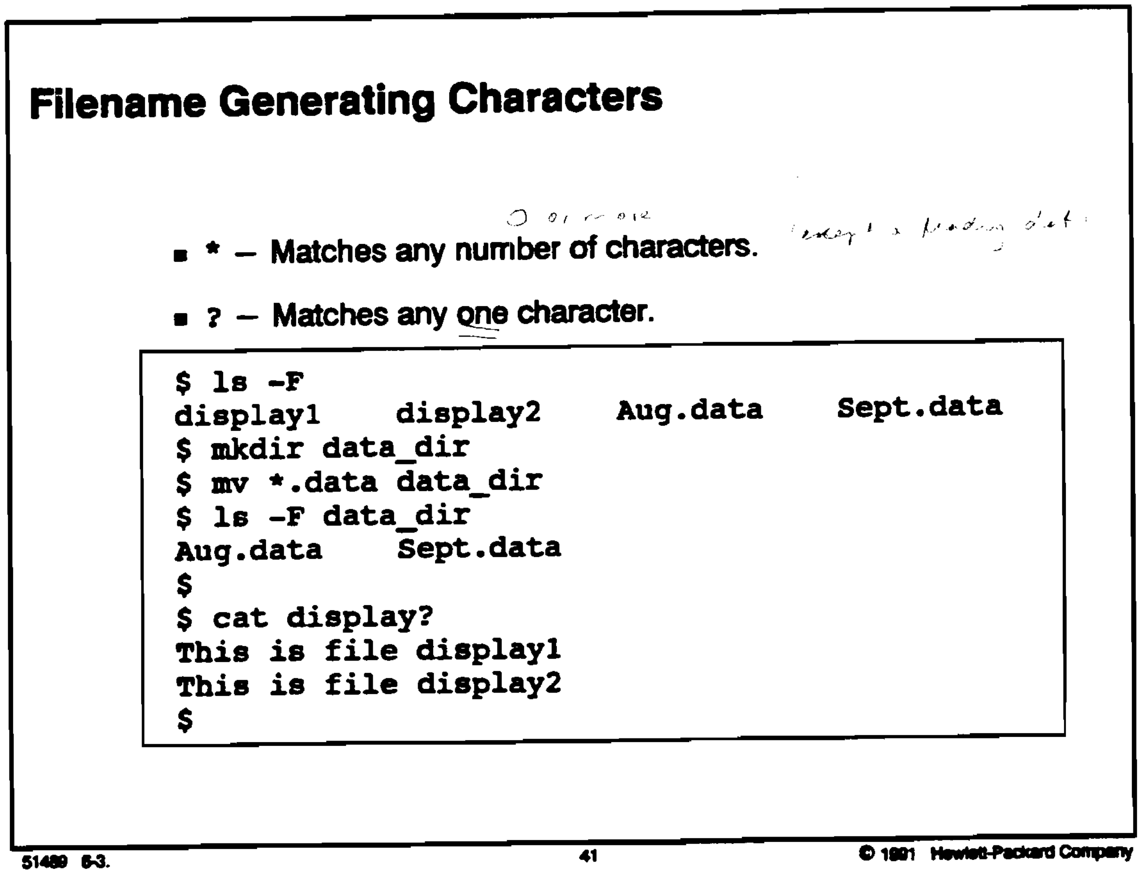
Slide Notes

Explain each command line and its corresponding output. Note that `file3` is really a directory. This is done to show that the name is just a name. It has no relation to the contents unless the user sets it up that way.

Transition

We can also use naming conventions to organize our files.

6-3. SLIDE: Filename Generating Characters



Filename Generating Characters

0 or more

- * – Matches any number of characters. (except a leading dot)
- ? – Matches any one character.

```
$ ls -F
display1    display2    Aug.data    Sept.data
$ mkdir data_dir
$ mv *.data data_dir
$ ls -F data_dir
Aug.data    Sept.data
$
$ cat display?
This is file display1
This is file display2
$
```

51489 6-3. 41 © 1991 Hewlett-Packard Company

Student Notes

Besides using directories, we can use different file naming conventions to help us organize our files. For example, if we were collecting data and storing the information in files, we could name all of our data files with a `.data` extension. Programming compilers use similar naming conventions. For example, C program source code files end with `.c`, Pascal with `.p`, and so on.

Suppose we wanted to move all of our data files to a new directory called `data_dir`. The slide shows how this can be done.

As you may have guessed, the "*" is a filename generating character that means match all characters. Thus, `*.data` means all file names ending with `.data`, regardless of the first part of the name.

A less common filename generating character is the "?". This means to match any one character. For example, in your home directory are two files named `display1` and `display2`. You can operate on these two files at the same time using the "?" filename generating character as shown on the slide.

Note

Neither the "*" nor "?" will match a leading ".". This indicates a dot file, and dot files are meant to be hidden, even from filename generating characters.

Caution

Be very careful when using filename generating characters with commands such as `rm`! If you accidentally remove a file, it is gone forever unless your administrator has made a backup copy.

Note that neither of these filename generating characters will be effective unless you enforce some sort of file naming convention in your directories.

6-3. SLIDE: Filename Generating Characters

Instructor Notes

Purpose

The filename generating characters presented here are a very valuable tool when you are using good file naming conventions.

Key Points

"*" matches any number of characters. "?" matches any one character.

Evaluation Question

How can I operate on all of my files in the current directory at once? Simply use "*"
(`cp * new_dir; rm *`).

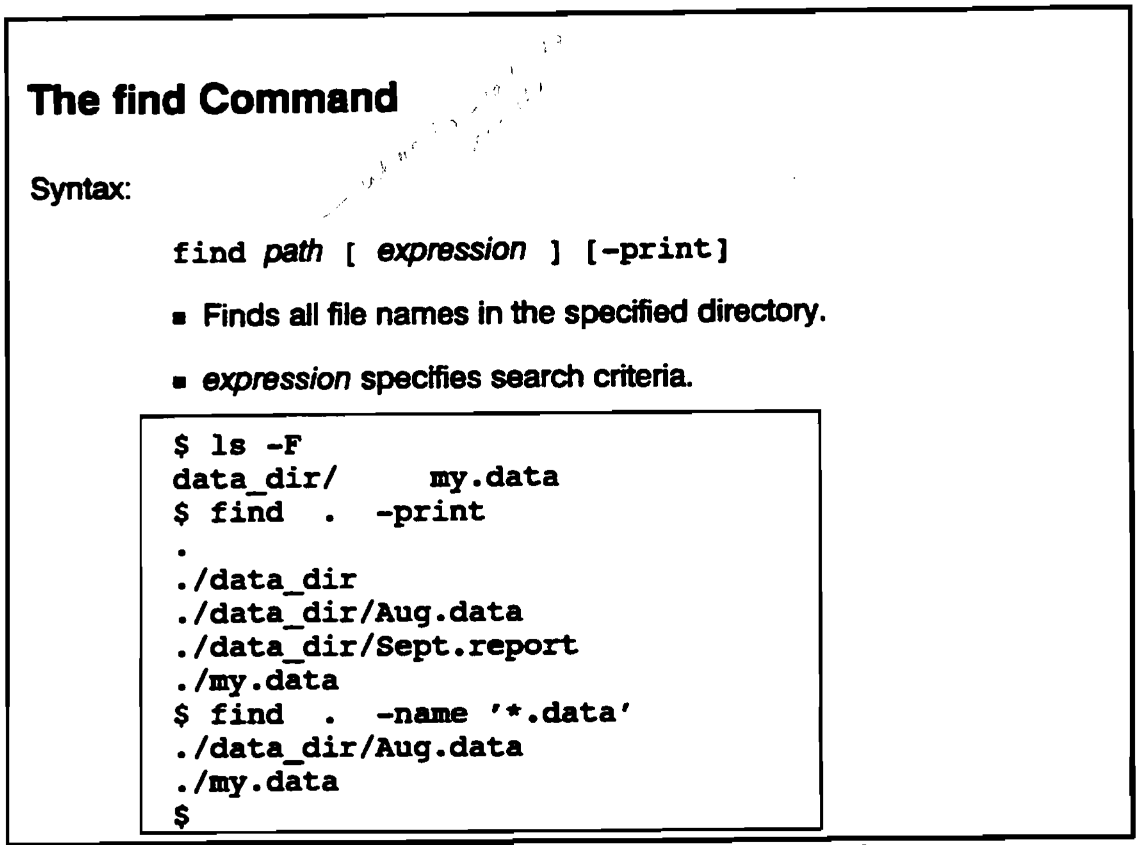
Where Problems Arise

Most students who are from non-UNIX system environments have trouble with the concept that "." is just another character in a filename. Thus `file1.data` is a legal name. In a system such as MS-DOS, this filename would not be legal. `.data` are just five more characters.

Transition

How can we find our files once we have them stored in directories?

6-4. SLIDE: The find Command



The find Command

Syntax:

find *path* [*expression*] [-print]

- Finds all file names in the specified directory.
- *expression* specifies search criteria.

```
$ ls -F
data_dir/     my.data
$ find  .  -print
.
./data_dir
./data_dir/Aug.data
./data_dir/Sept.report
./my.data
$ find  .  -name '*.data'
./data_dir/Aug.data
./my.data
$
```

51480 6-4. 42 © 1991 Hewlett-Packard Company

Student Notes

With all of the directories that we have to store our files in, how can we find a file if we do not remember which directory it is in?

The UNIX operating system has the `find` command to help us. `find` searches for filenames starting at a specified directory downward until it runs out of directories to search through. The `find` command has this general form that we will use:

`$ find` *directory* [*expression*] `[-print]`

where *directory* is the directory where we will start the recursive search, *expression* is an optional list of qualifiers we can use to restrict our search, and `-print` means print the filenames that are found. If `-print` is omitted, it is assumed.

Here are a few examples:

```
$ find . -print
.
./display1
./display2
./new.jersey
./add.text
./cars+dogs
./cars+dogs/car.models
./cars+dogs/car.models/ford
./cars+dogs/car.models/ford/sports
./cars+dogs/car.models/ford/sports/mustang
      ⋮
```

The dots at the end indicate the output continues. In this example, we are searching starting at the current directory ("."), finding all filenames below the current directory, and printing those names.

If we want to find only files with a certain name, we can use the `-name` option:

```
$ find cars+dogs -name 'mustang' -print
cars+dogs/car.models/ford/sports/mustang
$
```

We can also use the filename generating characters that we just discussed to find all files starting with a "ch" in the `/bin` directory:

```
$ find /bin -name 'ch*' -print
/bin/chgrp
/bin/chmod
/bin/chown
$
```

There are many more options available with the `find` command. See find(1) for more information.

6-4. SLIDE: The find Command

Instructor Notes

Purpose

It is easy to "lose" files in the file system. `find` gives us a way to locate them again.

Key Points

`find` does not require the `-print` option; this is the default action.

When using filename generating characters, make sure you use quotation marks so the shell will not interpret them first.

Evaluation Question

What would `find / -print` display? All files on the file system. If several students try this at once, the disk performance will slow down drastically.

Slide Notes

Use "reveal" to show the examples.

Transition

We have seen many ways of manipulating files. The `ls -l` (`ll`) command provides us with more information about the file itself.

6-5. SLIDE: The Long Listing

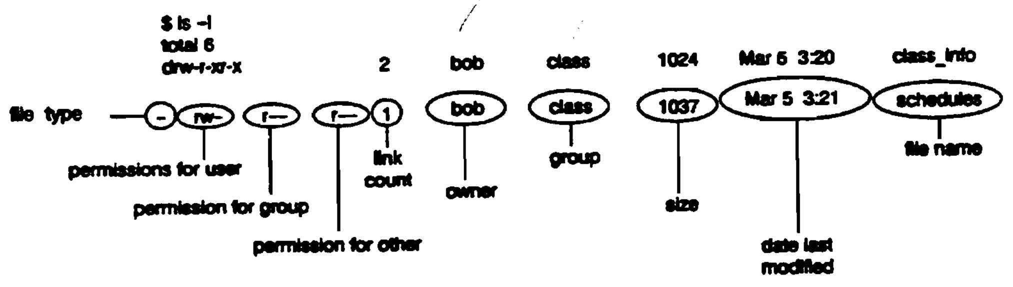
The Long Listing

Syntax:

```
ls -l
```

= ll

- Lists information for files in your current directory.



51480 6-5. 43 © 1991 Hewlett-Packard Company

Student Notes

The `ls -l` command produces a long listing of the directory contents.

There is quite a bit of information given in a long listing. We will take the first few lines displayed by the system on the slide as a sample.

`total 6`	The total number of 512-byte blocks of storage used by files and directories in this listing.
`-`	The type of file. A "d" indicates that this file is a directory. A "-" in this position indicates an ordinary file. There are a few other file types that need not concern us.
`rwxr--r--`	The permissions on the file (who can do what to the file).
`1`	A count of the links to the file.
`bob`	The owner of the file.
`class`	The group to which the file belongs.
`1037`	The size of the file in characters.

`Mar 5 3:21`	The date and time when the file was last modified.
`schedules`	The name of the file.

See ls(1) for further reference.

6-5. SLIDE: The Long Listing

Purpose

To see file characteristics, including file size, ownership, type, and last modification date/time.

Key Points

Use of the `-l` ("ell") option to the `ls` command provides information about files that a user is interested in seeing.

Presentation Suggestions

Explain each "field" in the output of `ls -l`.

The `ls -l` command is used so often that HP has developed an abbreviation for it. Typing `ll` is equivalent to typing `ls -l`.

Where Problems Arise

The "total" report is vendor-implementation dependent. For example, on all HP-UX machines, the smallest amount of space a file can take is one fragment (usually 1 kilobyte). Thus, even if a file's size is 1 byte, it will be reported as taking up two 512-byte blocks. This is why the slide shows a total of 6 (2 for the directory and 4 for the file).

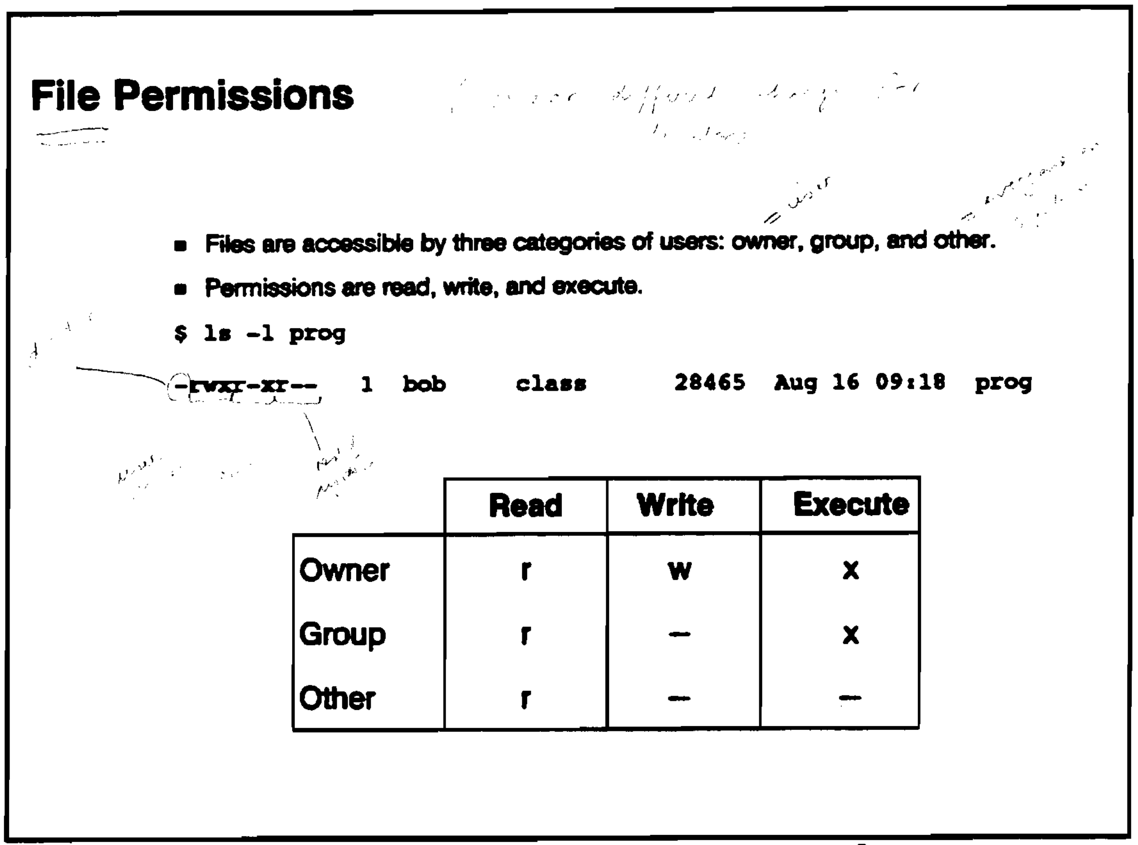
Slide Notes

Use the "reveal" method and explain the command line and output.

Transition

After seeing the command to print the characteristics of files, let's take a closer look at the very important file permissions.

6-6. SLIDE: File Permissions



File Permissions

[illegible]

- Files are accessible by three categories of users: owner, group, and other.
- Permissions are read, write, and execute.

```
$ ls -l prog
-rwxr-xr--   1  bob    class     28465  Aug 16 09:18  prog
```

	Read	Write	Execute
Owner	r	w	x
Group	r	–	x
Other	r	–	–

51489 6-6. 44 © 1991 Hewlett-Packard Company

Student Notes

Permissions specify who can do what to the file. Only the owner of a file can change its permissions. There are three fields of three characters that specify the permissions:

- The first field specifies the permissions for the owner (user) of the file.
- The second field specifies the permissions for the group to which the file belongs.
- The third field specifies the permissions for everyone else (other).

In each field, an r as the first character specifies read permission, a w specifies write permission, and an x specifies execute permission. A dash ("-") in any of the three positions indicates that the permission is turned off. These characters determine the permissions on a file:

Permission	Read	Write	Execute
Allowed	r	w	x
Denied	-	-	-

When you log in, the system knows your user id and sets your group id to what the system administrator has assigned for you. Every time you access a directory or file, the permissions on that directory or file are checked against your user and group id.

If you own the file, the user permissions are checked. If you are not the owner but you are a member of the group that the file belongs to, the group permissions are checked. If you do not own the file and you are not a member of the file's group, the other permissions are checked.

The superuser is a special login id referred to as **root.** Generally, only a few people on any UNIX system will know the root password. As the name implies, the superuser can access any file, and do whatever he or she wants, no matter what permissions are set on a file or directory.

The system administrator requires superuser status to install new users, new software, and fulfill the system maintenance and backup requirements. Thus, even if a file has no read permission, it will still be backed up by the system administrator, and can be restored in case of a system failure.

6-6. SLIDE: File Permissions

Instructor Notes

Purpose

Sometimes you cannot get access to a file, and you should understand why. You may also want to protect your files and directories against unauthorized access.

Key Points

Files and directories have permissions that affect who can use them.

Slide Notes

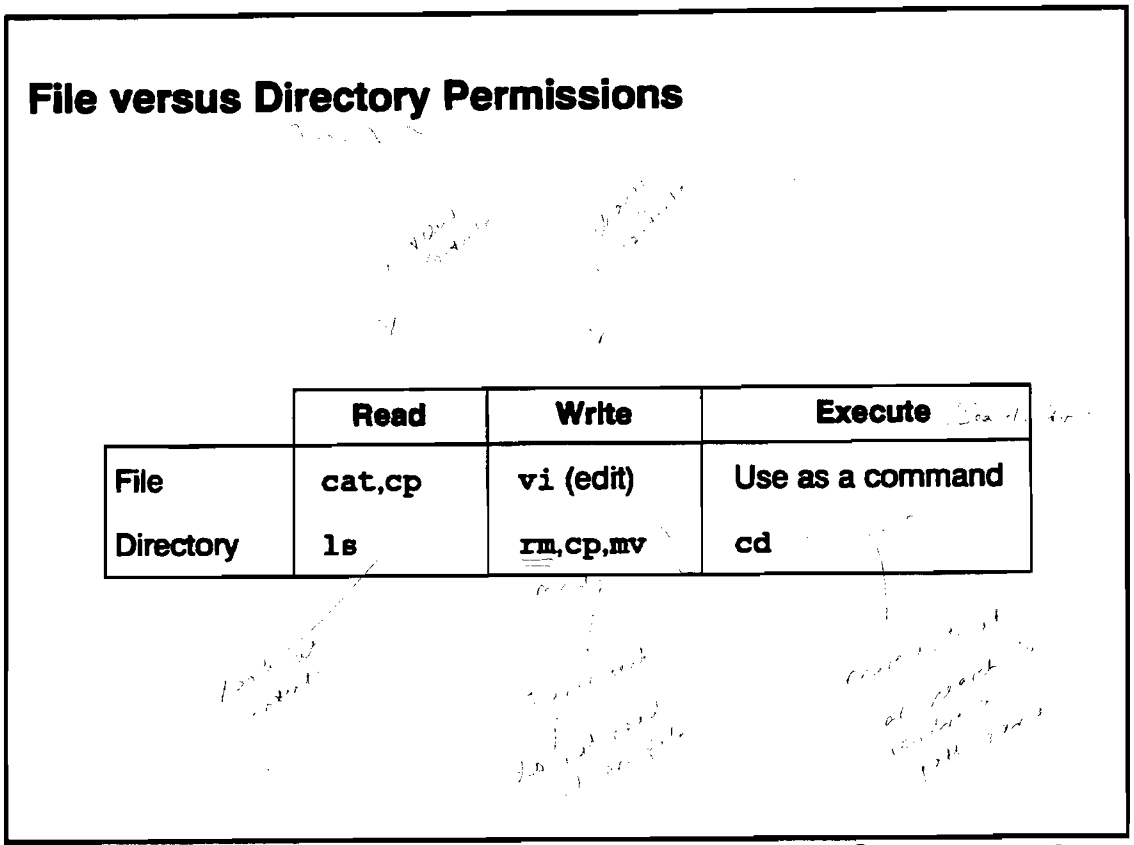
Ask who can read the file `prog` listed on the slide? Who can write it? Who can execute it?

Answer: Everyone can read the file; only `bob` (the owner) can write the file; only bob and the members of the group `class` can execute the file.

Transition

File permissions and directory permissions have slightly different meanings.

6-7. SLIDE: File versus Directory Permissions



File versus Directory Permissions

	Read	Write	Execute
File	`cat,cp`	`vi` (edit)	Use as a command
Directory	`ls`	`rm,cp,mv`	`cd`

51489 6-7. 45

Student Notes

Each permission has a different meaning when applied to a file or directory:

- Read permission on a file means that you can read its contents using `cp`, `cat`, and so on.
- Read permission on a directory means that you can `ls` the directory. The `ls` command looks at a directory's contents, in other words, the names of the files contained in it.
- Write permission on a file means that you can modify the contents of the file. Also, if you do not have write permission on a file and you try to remove it, you will be asked if you wish to proceed with the removal.
- Write permission on a directory means that you can remove or create files in the directory. Notice that to remove a file from a directory, you need only have write permission on the directory that contains it. You *do not* need write permission on the file itself; you will be prompted to proceed with the removal.
- Execute permission on a file means that you can use the file as a command. Commands such as `who` and `cat` are files with execute permissions set.

- Execute permission on a directory means that the directory may appear in a path name. Strictly speaking, execute permission means that the directory may be searched to resolve a path name.
- In order to obtain expected results when using `ls` or `ls -l` on a directory, you must have both read and execute permission on that directory.
- The default file permissions are `rw-rw-rw-`, and the default directory permissions are `rwxrwxrwx`.

6-7. SLIDE: File versus Directory Permissions

Instructor Notes

Purpose

Permissions on files have different meanings than directory permissions.

Key Points

Stress the fact that `rm` is a directory permission, *not* a file permission.

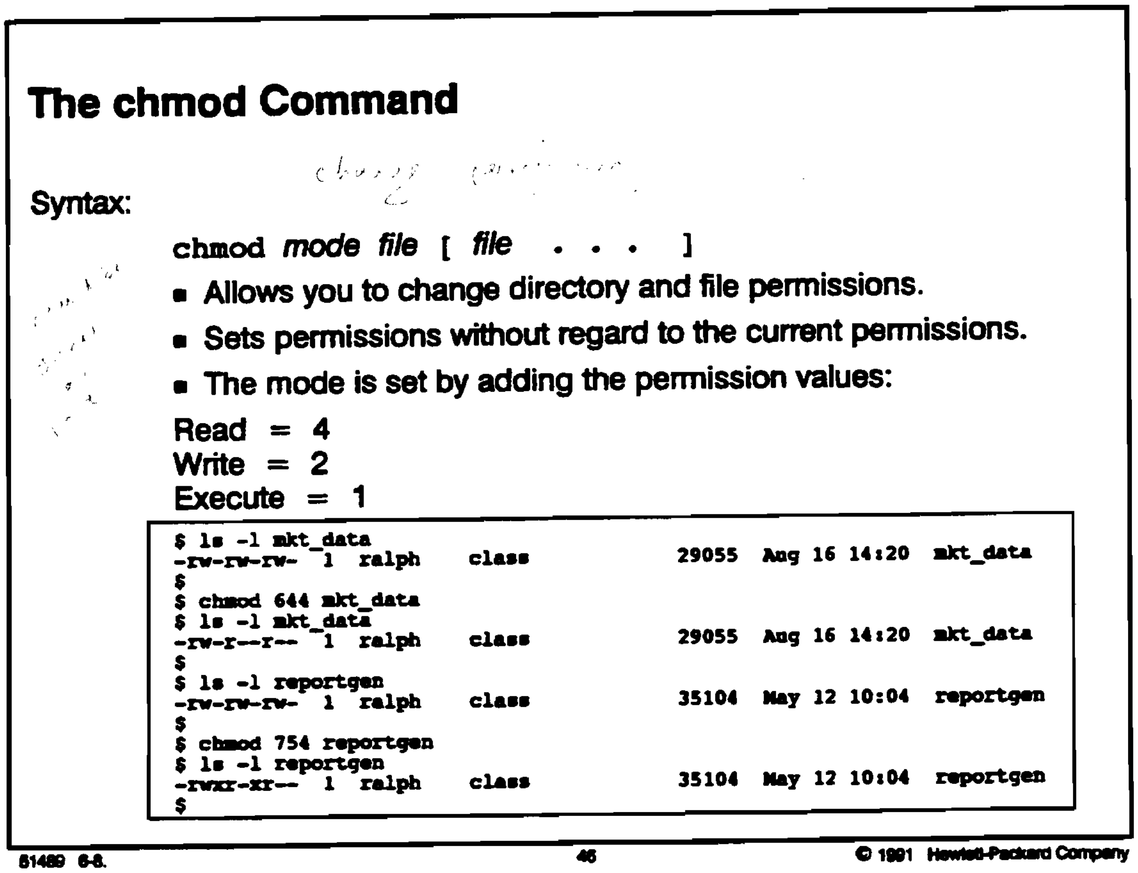
Presentation Suggestions

Some examples usually help at this point, especially with `rm`.

Transition

Let's look at how we can set and change permissions.

6-8. SLIDE: The chmod Command



The chmod Command

[illegible]

Syntax:

chmod *mode file* [*file* . . .]

- Allows you to change directory and file permissions.
- Sets permissions without regard to the current permissions.
- The mode is set by adding the permission values:

Read = 4
Write = 2
Execute = 1

```
$ ls -l mkt_data
-rw-rw-rw-  1  ralph    class          29055  Aug 16 14:20  mkt_data
$
$ chmod 644 mkt_data
$ ls -l mkt_data
-rw-r--r--  1  ralph    class          29055  Aug 16 14:20  mkt_data
$
$ ls -l reportgen
-rw-rw-rw-  1  ralph    class          35104  May 12 10:04  reportgen
$
$ chmod 754 reportgen
$ ls -l reportgen
-rwxr-xr--  1  ralph    class          35104  May 12 10:04  reportgen
$
```

51489 6-8. 46 © 1991 Hewlett-Packard Company

Student Notes

The chmod ("change mode") command is how we change the permissions (the "mode") of a file or directory. The command is in the following format:

chmod *mode file* [*file* ...]

where *mode* is the new permissions for the specified *file(s)*.

This form of the chmod command does not rely on what the current permissions are on a file. It simply uses three numbers to specify the new "mode." The first digit specifies the "user" permissions, the second is "group," and the third is "other." We calculate each of the three digits by figuring out what permissions we want set first. Then we simply add 4 to set read permissions, 2 for write, and 1 for execute.

For example, read and write permission for the user and read permission for both the group and other would be specified as 644. Read and write permission is 4 + 2 = 6; and just read permission is 4.

Note You must be the owner of the file in order to change its permissions.

6-8. SLIDE: The chmod Command

Instructor Notes

Purpose

Permissions define everything you can do on the system. We need a way to change permissions on our files to better secure and control them. This is a form of the chmod command that does not care what the current permissions are.

Key Points

It is easy to say r = 4, w = 2, and x = 1. The more advanced students will realize that this is really a set of three octal digits. Each one is derived from three bit positions (r, w, and x). The bit is 1 if the permission is set and 0 if it is not set. Thus, rwx is 111, which converts to a 7, and r-x is 101, which is 5.

Teaching Question

If the class is advanced, ask them if they can see how the numbers are really formed.

Presentation Suggestions

You may want to introduce another form of the chmod command that changes the existing permissions by adding or subtracting new permissions. It requires that you know the permissions in advance (usually by using the ls -l command). It uses u to define the user permissions, g to define the group permissions, and o to define the other permissions.

Here are some examples:

```
$ ls -l
-rw-rw-rw-   1 ralph   class         269 Feb 10  1988 myprog
-rw-rw-rw-   1 ralph   class       24976 Jun 21 10:13 module6
$
$ chmod go-w *
$ ls -l
-rw-r--r--   1 ralph   class         269 Feb 10  1988 myprog
-rw-r--r--   1 ralph   class       24976 Jun 21 10:13 module6
$
$ chmod +x myprog
$ ls -l myprog
-rwxr-xr-x   1 ralph   class        3891 Jun  9 14:54 myprog
$
```

Specifying go-w as the mode subtracted ("-") write permission (w) from the group and other permissions (go). We can add permissions with this form of the chmod command using "+". Notice, in the last example, who the permissions applied to is not specified, so chmod added ("+") execute permission ("x") in all three fields.

You can also use this form of chmod to set permissions without regard to the current permissions. This form uses the "=" operator. For example, the command

```
$ chmod ugo=rw module6
```

would set read and write permission (rw) for everyone (ugo). Recall that these are the default permissions for a file.

Slide Notes

Use "reveal" to explain and show the results of each command line.

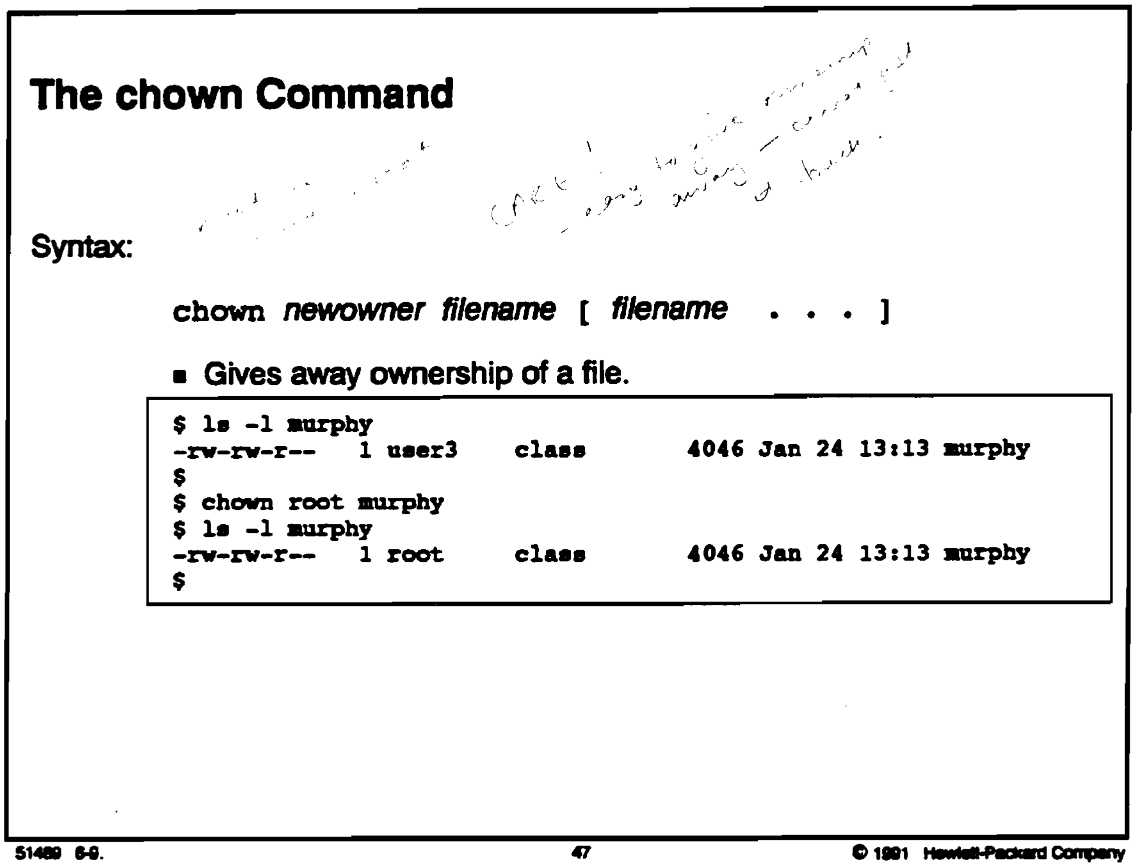
Chalkboard Notes

A few examples on the board might help to reinforce the concepts.

Transition

Let's look at how we can change the ownership of a file or directory.

6-9. SLIDE: The chown Command



The chown Command

[illegible]

Syntax:

chown *newowner filename* [*filename* . . .]

- Gives away ownership of a file.

```
$ ls -l murphy
-rw-rw-r--   1 user3    class      4046 Jan 24 13:13 murphy
$
$ chown root murphy
$ ls -l murphy
-rw-rw-r--   1 root     class      4046 Jan 24 13:13 murphy
$
```

51489 6-9. 47 © 1991 Hewlett-Packard Company

Student Notes

The owner of a file can choose to give away ownership of that file to another user on the system. This is accomplished by changing the ownership attribute of a file. To do this, we use the chown command. The syntax is:

chown *newowner filename* [*filename* ...]

where *newowner* is the username of the person who is to be the new owner of the file and *filename* is the name of the file that the owner wishes to give away. Only the owner of the file and the super-user may change the ownership of a file. The new owner specified must be a valid user on the system.

Once the ownership has been changed, only the new owner or the super-user can change it back.

[illegible]

6-9. SLIDE: The chown Command

Instructor Notes

Purpose

We may sometimes need to change the ownership of a file or directory.

Key Points

Stress that only the owner may change ownership of a file or directory.

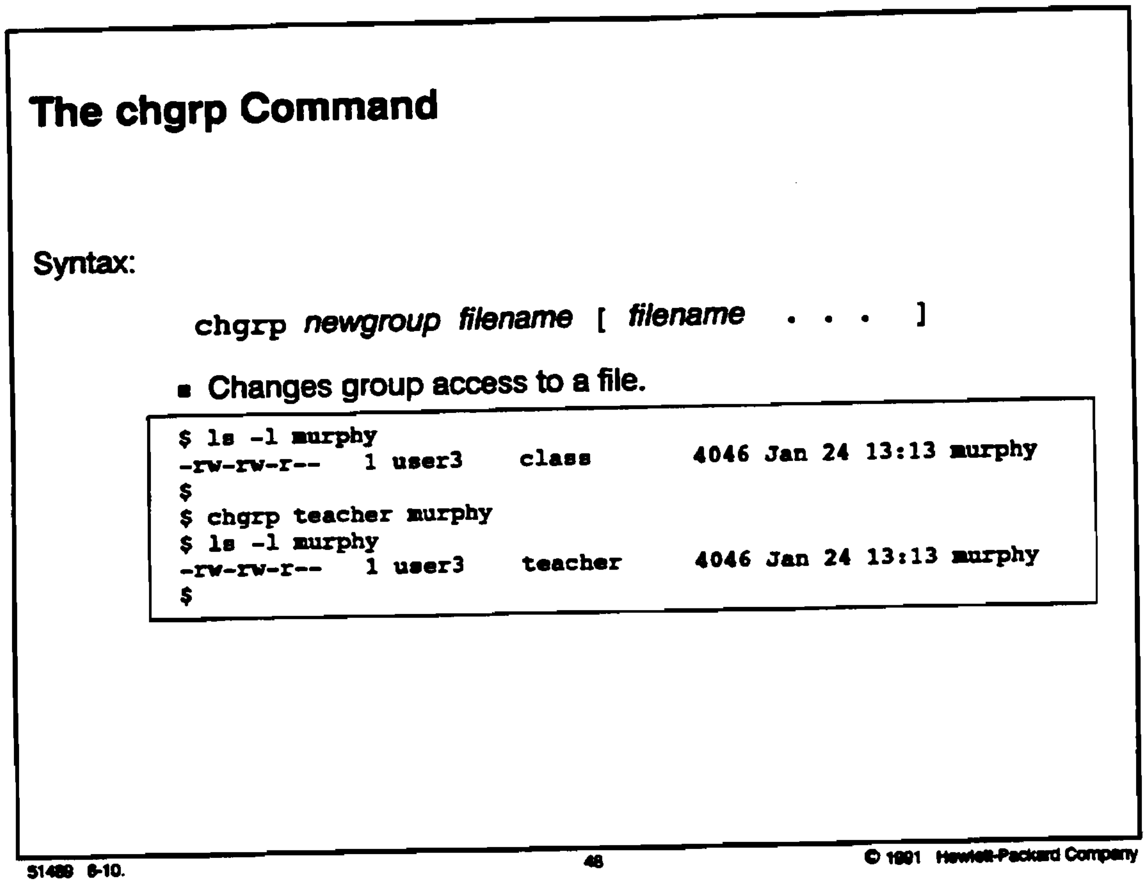
Where Problems Arise

Some students forget that once they give away ownership of a file or directory they cannot get it back unless the new owner wants to give it back.

Transition

Let's look at how we can change the group membership of a file or directory.

6-10. SLIDE: The chgrp Command



The chgrp Command

Syntax:

chgrp *newgroup filename* [*filename* . . .]

- Changes group access to a file.

```
$ ls -l murphy
-rw-rw-r--   1 user3    class        4046 Jan 24 13:13 murphy
$
$ chgrp teacher murphy
$ ls -l murphy
-rw-rw-r--   1 user3    teacher      4046 Jan 24 13:13 murphy
$
```

51489 6-10. 48 © 1991 Hewlett-Packard Company

Student Notes

The group access attribute of a file can be changed as readily as the ownership attribute of the file. This is accomplished with the chgrp command. The syntax is:

chgrp *newgroup filename*

Where *newgroup* is the name of the group that will now have access (through the group permissions) to the file called *filename*. Changing the group access *does not* affect the owner of the file. The ownership attribute remains the same.

You do not have to belong to the group in order to do a chgrp. However, the chgrp command will not work if the new group specified does not exist. Group existence and membership is controlled by the system administrator.

Again, only the owner of a file or the super-user can change the group access.

6-10. SLIDE: The chgrp Command

Instructor Notes

Purpose

We may sometimes need to change the group membership of a file or directory.

Key Points

Stress that only the owner may change the group membership of a file or directory.

Where Problems Arise

Remind students that the group they wish to change to must already exist.

Transition

The next topic is an exercise.

6-11. LAB: Exercises

1. Create two directories under your home directory named `mail` and `junk`. Obtain a long listing of the contents of the two directories with the `ls -al` command. Note what files you see.

2. Copy `display1` and `display2` into the `junk` directory. Change the permissions of `junk` so that you do not have read permission. Verify that the new permissions have been set. Obtain a long listing of `junk`. What happens?

Advanced:

There is an option to the `ls -l` command that makes it easy to list just the information about a particular directory without listing its contents. Try this option on the `junk` directory.

3. Change the permissions of `junk` so that you have read and write permission, but no execute permission. Obtain a long listing of `junk`. What happens?

4. Change the permissions of `junk` back to their default permissions. Change the permissions of `display1` in the `junk` directory so that you do not have write permission on the file. Remove `display1` from the `junk` directory. What happens?

5. Remove the junk directory.

6. Obtain a long listing of the files in your home directory staring with the letter "d."

7. Obtain a listing of all files in your home directory with the letter "t" somewhere in their names.

8. What is the command to change the ownership of `cbt` to your mail partner? Do it now. Can you get ownership back? How?

9. From your home directory, copy the file `mutant` to the directory `/bin`. Did you have any problems? If so, why?

10. Copy the file `/bin/cp` to your home directory. Did you have any problems? If so, why? What command did you use?

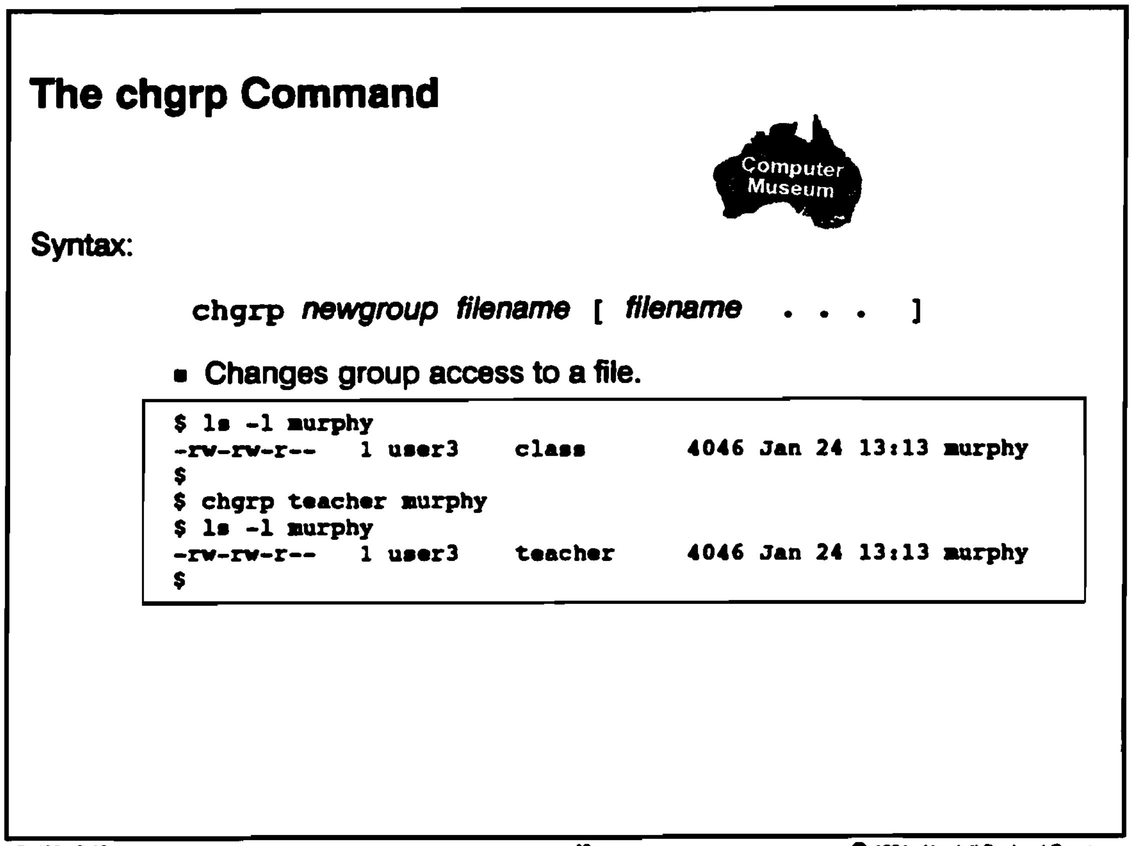
Advanced Exercises

1. List the names of all files in `/usr/spool` that belong to the user named `lp`. (You will have to look in find(1) for this one).

2. If two files are linked, they share the same space on the disk. This space is referenced by an "inode number". Use the `ln` command to create a link between two files, and then use the `-i` option of `ls` to see that the files really have the same inode number.

3. You see that a file is linked to another file. How can you find the name of the other file? (Hint: There is a find(1) option that will help.)

6-10. SLIDE: The chgrp Command



The chgrp Command

Syntax:

chgrp *newgroup filename* [*filename* . . .]

- Changes group access to a file.

```
$ ls -l murphy
-rw-rw-r--   1 user3    class       4046 Jan 24 13:13 murphy
$
$ chgrp teacher murphy
$ ls -l murphy
-rw-rw-r--   1 user3    teacher     4046 Jan 24 13:13 murphy
$
```

51489 6-10.   46   © 1991 Hewlett-Packard Company

Student Notes

The group access attribute of a file can be changed as readily as the ownership attribute of the file. This is accomplished with the chgrp command. The syntax is:

chgrp *newgroup filename*

Where *newgroup* is the name of the group that will now have access (through the group permissions) to the file called *filename*. Changing the group access *does not* affect the owner of the file. The ownership attribute remains the same.

You do not have to belong to the group in order to do a chgrp. However, the chgrp command will not work if the new group specified does not exist. Group existence and membership is controlled by the system administrator.

Again, only the owner of a file or the super-user can change the group access.

[illegible]

6-11. LAB: Exercises

1. Create two directories under your home directory named `mail` and `junk`. Obtain a long listing of the contents of the two directories with the `ls -al` command. Note what files you see.

2. Copy `display1` and `display2` into the `junk` directory. Change the permissions of `junk` so that you do not have read permission. Verify that the new permissions have been set. Obtain a long listing of `junk`. What happens?

Advanced:

There is an option to the `ls -l` command that makes it easy to list just the information about a particular directory without listing its contents. Try this option on the `junk` directory.

3. Change the permissions of `junk` so that you have read and write permission, but no execute permission. Obtain a long listing of `junk`. What happens?

4. Change the permissions of `junk` back to their default permissions. Change the permissions of `display1` in the `junk` directory so that you do not have write permission on the file. Remove `display1` from the `junk` directory. What happens?

5. Remove the junk directory.

6. Obtain a long listing of the files in your home directory staring with the letter "d."

[illegible]

7. Obtain a listing of all files in your home directory with the letter "t" somewhere in their names.

[illegible]

8. What is the command to change the ownership of cbt to your mail partner? Do it now. Can you get ownership back? How?

[illegible] cbt

[illegible]

9. From your home directory, copy the file mutant to the directory /bin. Did you have any problems? If so, why?

[illegible]

[illegible]

[illegible]

10. Copy the file /bin/cp to your home directory. Did you have any problems? If so, why? What command did you use?

NO

[illegible]

[illegible]

Advanced Exercises

1. List the names of all files in **/usr/spool** that belong to the user named **lp**. (You will have to look in find(1) for this one).

[illegible]

2. If two files are linked, they share the same space on the disk. This space is referenced by an "inode number". Use the **ln** command to create a link between two files, and then use the **-i** option of **ls** to see that the files really have the same inode number.

[illegible]

3. You see that a file is linked to another file. How can you find the name of the other file? (Hint: There is a find(1) option that will help.)

6-11. LAB: Exercises

Transition

We have seen many commands to list filenames, to copy, rename, and remove files, and to change the permissions of files. In the next module we will learn how to create and edit text files using the `vi` editor.

Solutions

1. Create two directories under your home directory named `mail` and `junk`. Obtain a long listing of the contents of the two directories with the `ls -al` command. Note what files you see.

Answer:

```
$ cd
$ mkdir mail junk
$ ls -al mail junk

junk:
total 4
drwxrwxrwx   2 bob    aeo         64 Jun 16 11:01 .
drwxrwxrwx   6 root   other     1024 Jun 16 11:01 ..

mail:
total 4
drwxrwxrwx   2 bob    aeo         64 Jun 16 11:01 .
drwxrwxrwx   6 root   other     1024 Jun 16 11:01 ..
```

Notice that the `mkdir` command automatically created "." and ".." in each directory. The `-a` option to `ls -l` allows us to view file and directory names with a leading ".".

2. Copy `display1` and `display2` into the `junk` directory. Change the permissions of `junk` so that you do not have read permission. Verify that the new permissions have been set. Obtain a long listing of `junk`. What happens?

Advanced:

There is an option to the `ls -l` command that makes it easy to list just the information about a particular directory without listing its contents. Try this option on the `junk` directory.

Answer:

```
$ cp display1 display2 junk
$ chmod 377 junk
$ ls -l junk
junk unreadable
total 2
$ ls -ld junk
```

```
d-wxrwxrwx   2 bob      aeo           1024 Jun 21 14:50 junk
$
```

Since you do not have read permission on the directory, you cannot look at the contents of the directory.

Notice that the `-d` option allows us to view a directory itself instead of its contents.

`chmod u-r junk` would also have worked.

3. Change the permissions of `junk` so that you have read and write permission, but no execute permission. Obtain a long listing of `junk`. What happens?

Answer:

```
$ chmod 677 junk
$ ls -l junk
display1/junk not found
display2/junk not found
total 0
```

Since you do not have execute permission on the directory, you cannot search the directory.

4. Change the permissions of `junk` back to their default permissions. Change the permissions of `display1` in the `junk` directory so that you do not have write permission on the file. Remove `display1` from the `junk` directory. What happens?

Answer:

```
$ chmod 777 junk
$ cd junk
$ chmod 466 display1
$ rm display1
display1: 466  mode ? (y/n) y
```

Since you do not have read permission on the file, you are prompted to proceed with the removal.

5. Remove the junk directory.

Answer:

```
$ rm display2
$ cd ..
$ rmdir junk
$
```

Note that since `junk` was not empty, you first had to remove the files that it contained. Note that `rm -r junk` would have worked also.

6. Obtain a long listing of the files in your home directory staring with the letter "d."

Answer:

```
$ ls -l d*
-rw-rw-r--    1 bob    aeo                    204 Mar 17  1988 delete
-rw-rw-r--    1 bob    aeo                     22 Mar 17  1988 display1
-rw-rw-r--    1 bob    aeo                     22 Mar 17  1988 display2
$
```

7. Obtain a listing of all files in your home directory with the letter "t" somewhere in their names.

Answer:

```
$ ls -l *t*
-rw-rw-r--    1 bob    aeo                  35 Mar 17  1988 add.text
-rw-rw-r--    1 bob    aeo                 172 Jan 10 16:11 cbt
-rw-rw-r--    1 bob    aeo                 204 Mar 17  1988 delete
-rw-rw-r--    1 bob    aeo                 127 Mar 17  1988 mutant
-rw-rw-r--    1 bob    aeo                6842 Mar 17  1988 unwanted
```

8. What is the command to change the ownership of `cbt` to your mail partner? Do it now. Can you get ownership back? How?

Answer:

$ chown *mail_partner_login_id* cbt

You can only get ownership back if your mail partner gives it back to you.

9. From your home directory, copy the file `mutant` to the directory `/bin`. Did you have any problems? If so, why?

Answer:

The `/bin` directory does not have *write* permission set for others.

10. Copy the file `/bin/cp` to your home directory. Did you have any problems? If so, why? What command did you use?

Answer:

You should have no problems. The *read* permissions are set for others on `/bin/cp`. The following command would have performed the copying function:

```
$ cp /bin/cp .
```

Advanced Exercises

1. List the names of all files in /usr/spool that belong to the user named lp. (You will have to look in find(1) for this one).

Answer:

```
$ find /usr/spool -user lp -print
/usr/spool/lp
/usr/spool/lp/class
/usr/spool/lp/interface
     :
```

2. If two files are linked, they share the same space on the disk. This space is referenced by an "inode number". Use the `ln` command to create a link between two files, and then use the `-i` option of `ls` to see that the files really have the same inode number.

Answer:

```
$ ln display1 linkfile
$ ls -li display1 linkfile
105667 -rw-rw-r--   2 bob    aeo                  22 Mar 17 1988 display1
105667 -rw-rw-r--   2 bob    aeo                  22 Mar 17 1988 linkfile
$
```

105667 is the inode number that both files share. Notice also that the link count for each file is 2.

3. You see that a file is linked to another file. How can you find the name of the other file? (Hint: There is a find(1) option that will help.)

Answer:

Find a file with the same inode number:

```
$ ls -i display1
105667  display1
```

Execute `find` using the `-inum` option to find the other names that reference this file:

```
$ find . -inum 105667 -print
./display1
./linkfile
$
```

Note that this is a simple case. It becomes a much bigger problem if files are linked in several directories.

Module 7 — The vi Editor

Objectives

Upon completion of this module, you will be able to do the following:

- Explain the three modes of `vi`.
- Enter and exit the `vi` editor.
- Use the cursor movement keys in `vi`.
- Use the `vi` editor to create and store a new text file.
- Use the appropriate `vi` commands to insert text.
- Use the appropriate `vi` commands to delete text.
- Use the appropriate `vi` commands to modify text.
- Use the `vi` editor to modify an existing text file.

Module 7 — The vi Editor

Overview of Module 7

This module is a minimal introduction to the `vi` editor. We chose `vi` because it is a powerful screen-oriented text editor that is readily available on almost all UNIX systems. By the end of the module, the student will be a functional user of the `vi` editor.

Do not try to do this module without a break. It will take between 2 and 3 hours to cover. Take a break after an hour or so.

Motivation

The students must know how to use an editor in order to write any programs or do any substantial work on their UNIX system.

Exercises

The exercises are dispersed throughout this module. Since learning to use `vi` is a skill, we feel it is important to immediately and continuously exercise the material. Please do all exercises for a topic when the exercises are presented.

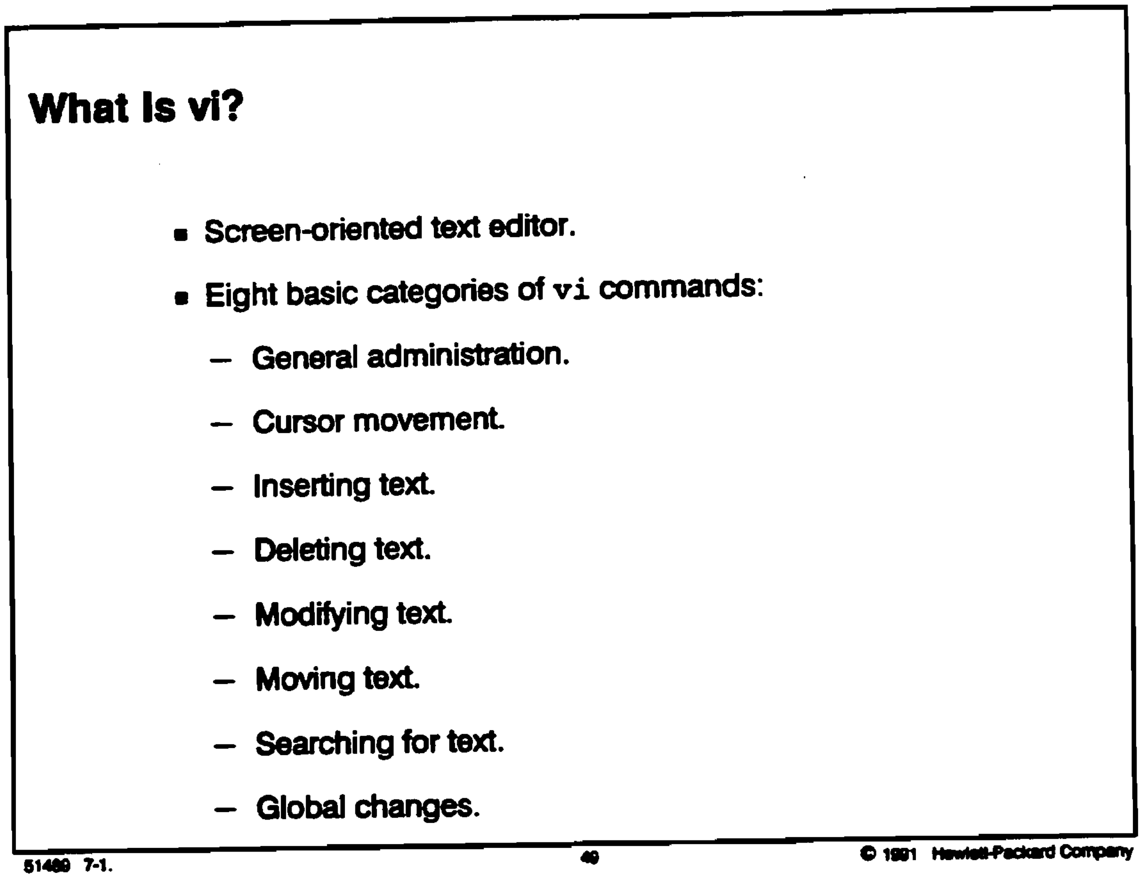
Exercise Files

`add.text`, **`delete`**, **`mutant`**, **`new.jersey`**, **`numbers`**, **`productivity`**, **`unwanted`**, **`upside_down`**

Module Transition

Once we know how to edit a file, we can use `vi` commands with some of the features of the Korn Shell.

7-1. SLIDE: What Is vi?



What Is vi?

- Screen-oriented text editor.
- Eight basic categories of `vi` commands:
  - General administration.
  - Cursor movement.
  - Inserting text.
  - Deleting text.
  - Modifying text.
  - Moving text.
  - Searching for text.
  - Global changes.

51469 7-1. 49 © 1991 Hewlett-Packard Company

Student Notes

`vi` (pronounced "vee-eye" meaning "visual") is a text editor. A text editor is an interactive computer program that you control to enter or change text in a file. You can use a text editor to create a new file or to modify an existing one.

`vi` was developed at the University of California at Berkeley by William Joy. It is a screen-oriented interactive editor. This means that you work on the file one screen at a time. As you make changes to a file, they are immediately shown on the screen.

The `vi` editor was designed to be terminal independent. It may or may not take advantage of terminal function keys, or other special terminal keys. Thus, `vi` can be used on any type of terminal, so you do not have to learn keys each time you sit at a different type of terminal. `vi` is powerful enough to allow you to customize your session to take advantage of a particular terminal.

`vi` is invoked like any other UNIX system command. Once you are in `vi`, you will have to use its own special commands. `vi` has a large set of commands that give you many ways to accomplish the same desired result.

What you will come away with in this module is basic `vi` literacy. Using `vi` is a skill. As with any skill, it takes practice, and the more you practice, the better you become. Using `vi` will get easier as you use it and will ultimately become something you use without too much thought. You will come away with a good foundation for adding to your skills at your own pace.

Refer to the *The Ultimate Guide to the vi and ex Text Editors* for more information.

7-1. SLIDE: What Is vi?

Instructor Notes

Purpose

To introduce the topic of `vi` to the students and begin to get them thinking of editing text.

Key Points

`vi` is a screen-oriented text editor that has its own subset of sophisticated commands to manipulate text.

This course was written using `vi` and an advanced publishing package.

Teaching Question

Before you reveal the categories of `vi` commands, ask the students what they would like an editor to do.

Positive Instance

List students' thoughts on what an editor should do on a flip chart. Then, throughout the module, you can relate the `vi` commands to the students' thoughts of the "ideal" editor.

Where Problems Arise

This is very straightforward and there should be no problems. Make sure that the students are aware that `vi` has its own set of commands. They will need to learn the new commands that `vi` supports.

Evaluation Questions

What is a screen-oriented text editor and what would make it sophisticated?

Presentation Suggestions

Use "reveal" on the slide.

Transition

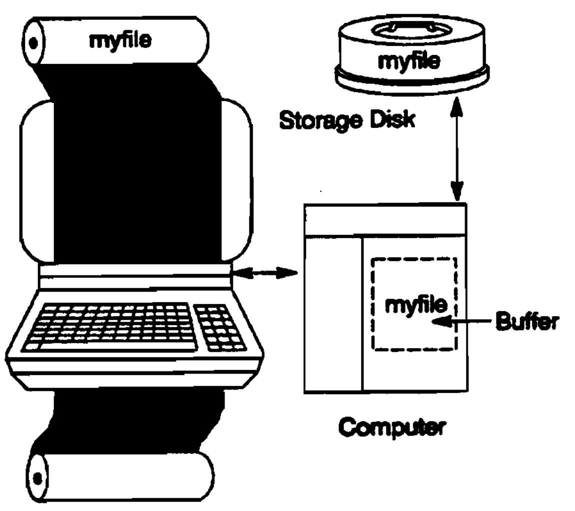
What do we first need to do to use the `vi` editor?

7-2. SLIDE: The vi Command

The vi Command

Syntax:

```
        vi filename
            $vi myfile
```



- Gets the file for editing.
- Copies the file to a buffer.
- Editing is done in the buffer.
- Original file is changed only when the buffer is saved.

A Window into a File

51489 7-2. 50 © 1991 Hewlett-Packard Company

Student Notes

We invoke the `vi` editor with the command:

```
$ vi filename
```

This tells the computer to execute `vi` and to get the file called *filename* from the disk for editing.

If you wanted to edit a file called `new.jersey`, you would type

```
$ vi new.jersey
```

When `vi` gets a file to edit, it does not give you the original; it gives you a copy which it puts into a **buffer** (another temporary file). Your original will *not* be changed until you tell `vi` to save the buffer. As a result, if you damage the file, there is no harm done to the original contents of the file.

As you enter `vi`, your computer terminal becomes a window into the file you are editing. The lines displayed on your screen are called the **text window**. We will perform of all our editing in this text window.

The cursor is always on the screen somewhere; it stays in the text window. The cursor lets us know where an action is to take place.

Exercise

Go to your login directory. All files for this module are in that directory. This is an interactive lecture and lab. Please do not work ahead of the instructor.

Use `vi` to edit a file called **`new.jersey`**. Notice where the cursor appears and notice any messages you may get from `vi`.

Do not type anything else yet, please!

7-2. SLIDE: The vi Command

Instructor Notes

Purpose

Show the students the syntax of the `vi` command, what happens once the `vi` editor is invoked, and how a file is displayed. Note that we are dealing with a file that already exists; we are not creating the file.

Key Points

`vi` is a UNIX system command that takes a file name as an argument. `vi` uses buffers so the original file is not touched until you tell `vi` to change it.

`vi` acts like a window into our file. All editing is done in this window.

Teaching Question

What is a buffer? Answer: A temporary file.

Suppose we have a file that is over 24 lines. Our terminal is only capable of 24 lines of text. How does `vi` handle this?

Answer: It uses the text window.

Where Problems Arise

Students will not have trouble with the idea of `vi` being a command, but they may need a little more time on the concept of a buffer. If they are a sharp group, you might even tell them that the buffer is really a temporary file called **`/tmp/Ex$$`**.

Evaluation Questions

Can we operate on parts of the file that are not displayed on our screen? Usually we cannot, but some of the global commands can do this.

Slide Notes

Explain each point in order, making sure that you spend enough time on the concept of a buffer.

Chalkboard Notes

The board is there to use for diagrams if the idea of buffer is difficult to grasp. You might draw a box for the file and then show how we invoke `vi` and then draw another box to show a buffer file is made that is an exact duplicate. Then show that we work in the buffer box. Scribble in some changes into the buffer box. Note that the boxes are now different. Then we save; the original box is brought up to date with the buffer box. Make the same scribbles in both boxes and say that both boxes are again the same.

Answer to Exercise

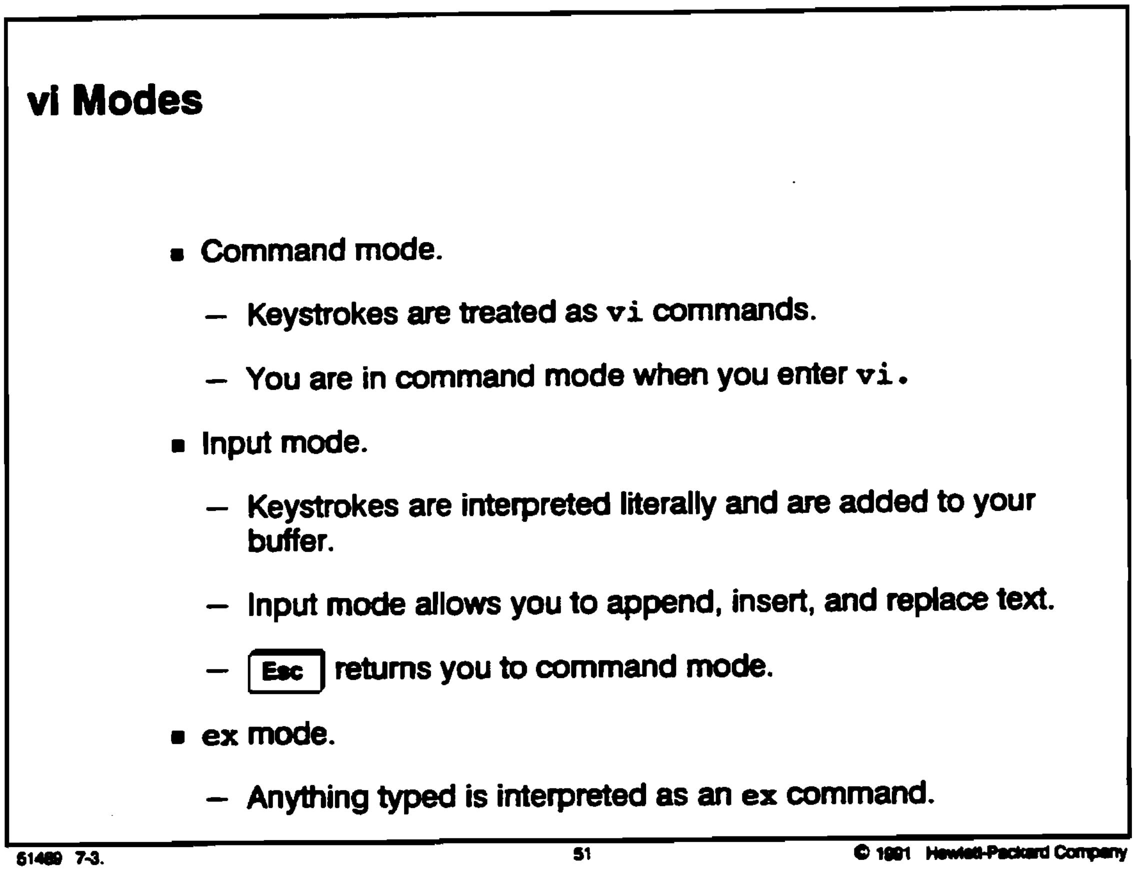
```
$ vi new.jersey
```

The first screen of text is displayed on your screen, and the cursor is on the first nonblank character of the first line. You will also get a message that indicates the size of the file at the bottom of the screen.

Transition

`vi` really has three modes in which we operate.

7-3. SLIDE: vi Modes



vi Modes

- Command mode.
  - Keystrokes are treated as `vi` commands.
  - You are in command mode when you enter `vi`.
- Input mode.
  - Keystrokes are interpreted literally and are added to your buffer.
  - Input mode allows you to append, insert, and replace text.
  - Esc returns you to command mode.
- ex mode.
  - Anything typed is interpreted as an ex command.

51489 7-3. 51 © 1991 Hewlett-Packard Company

Student Notes

`vi` has three "modes," that is, it interprets what you type in three different ways, depending on the current mode. The initial mode when you first enter `vi` is **command mode.** In this mode, `vi` interprets what you type at the keyboard to be a `vi` command. The command will not be printed on your screen, but you will see the action take place.

Input mode allows you to type text into the buffer. Your terminal acts like a typewriter, and *everything* you type is added to the buffer until you tell `vi` that you wish to return to command mode.

We will discuss several commands that will put you into input mode, but there is only one key that will return you to command mode, the Esc key. Thus, if you forget what mode you are in, you can always press Esc a few times until you hear a beep and then you will know for sure you are in command mode.

There are several commands to get to the **ex mode**, all of which start with a colon (:), slash (/), or question mark (?). The **ex** editor is the line-oriented counterpart of `vi`.

While you are in `vi`, you can type `:set showmode` Return. This will display `INPUT MODE` at the bottom of your screen when you are in input mode. If this message is not there, you must be in command mode.

`:set noshowmode` Return turns off this reminder.

7-3. SLIDE: vi Modes

Instructor Notes

Purpose

A thorough understanding of these three modes is critical to understanding `vi`. It is one of the frustrations for a first-time user.

Key Points

`vi` has three modes, command mode, input mode, and ex mode. Most other editors have only one mode and use special control characters to perform the functions of the `vi` commands.

Teaching Question

How do you know what mode you are in?

Answer: Do a `:set showmode` Return. If `INPUT MODE` is displayed at the bottom of the screen, then you are in input mode. Otherwise, you are in command mode.

Where Problems Arise

There is always confusion over command mode and input mode. Just assure the students that the hardest part of learning `vi` is remembering what mode they are in.

Evaluation Questions

If the characters that I am typing appear on the screen, what mode am I in?

Answer: Input mode.

Presentation Suggestions

Use "reveal" and explain each mode.

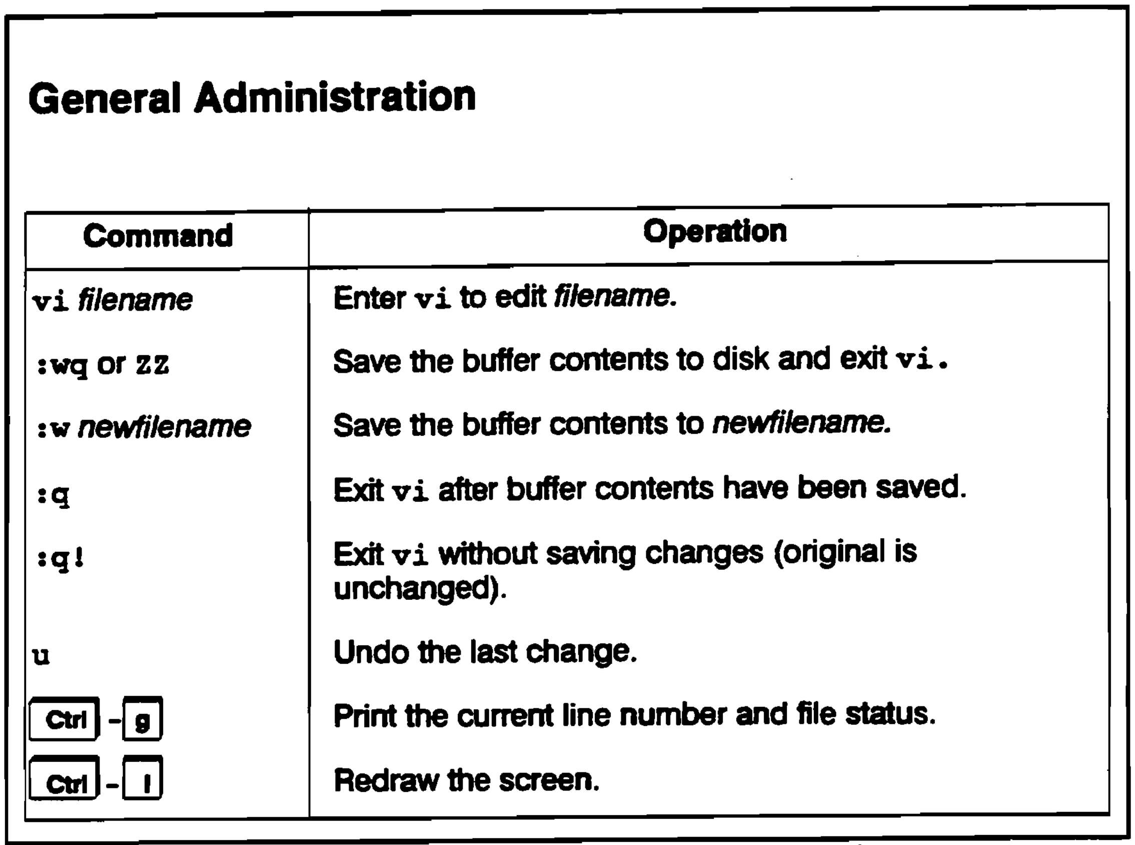
Slide Notes

Be sure to explain how characters are interpreted depending the mode. In input mode, the characters are added to the buffer, while in command mode they are interpreted as `vi` commands.

Transition

Let's cover some general `vi` administration topics.

7-4. SLIDE: General Administration



General Administration

Command	Operation
`vi` *filename*	Enter `vi` to edit *filename*.
`:wq` or `ZZ`	Save the buffer contents to disk and exit `vi`.
`:w` *newfilename*	Save the buffer contents to *newfilename*.
`:q`	Exit `vi` after buffer contents have been saved.
`:q!`	Exit `vi` without saving changes (original is unchanged).
`u`	Undo the last change.
Ctrl - g	Print the current line number and file status.
Ctrl - l	Redraw the screen.

51489 7-4. 52

Student Notes

Before we can talk about creating or editing files, we need to know a little about the administration commands of `vi`.

Entering vi

To enter `vi`, you will specify a file name for `vi` to edit. Once you are successfully entered into `vi`, you will be in command mode.

```
$ vi filename
```

Saving Your Work

After you have made some changes to a file, you need to tell `vi` when you are ready to save those changes. The command `:wq` or ZZ will save your text and return you to the shell; `:wq` and ZZ both mean "write my buffer and quit" `vi` (write and quit).

After `vi` has saved your file, it will print a message telling you the file name where your text was stored, possibly an informational message such as *New file*, and a count of the number of lines and characters that it saved. `vi` will then exit and you will be returned to your shell prompt.

If you want to save your changes under another name, you can give an argument `:w` *new_filename*. You would also use this form if you entered `vi` without using a file name argument.

Undoing Changes

There may come a time when you are editing a file and you "mangle" it through some accident of keystrokes. The u command (for undo) will undo the ***last change you made to the file***. If you decide that you liked the change, u will put it back. This means that u can undo itself! It simply undoes the last change.

If u cannot fix a large mistake, the way to return the file to its original state is to type `:q!`. Normally, `vi` will not let you quit until you have saved your changes. `:q!` means that you want to quit regardless of the fact that changes will be lost. The buffer is removed and you are free to `vi` the original file again.

vi Status

The Ctrl-g command displays the status of the file you are editing. This includes the file name, whether or not the file has been modified, and what line you are currently on in the file.

Redrawing the Screen

If for some reason your screen gets "mangled" while you are in `vi`, you can press Ctrl-l to redraw it.

7-4. SLIDE: General Administration

Purpose

To present the general administration commands used often in `vi`: `:wq`, `ZZ`, `:q!`, `u`, and Ctrl-g.

Key Points

`:wq` and `ZZ` are `vi`'s way of saving the buffer and exiting.

With `:q!`, you can discard the contents of the buffer without saving the contents.

Ctrl-l will redraw your screen if this becomes necessary (for example, someone wrote a message to your terminal).

Teaching Questions

What does saving the buffer mean? Answer: Writing the contents of the temporary file back out to the original file.

Am I working on my file when I am in `vi`? Answer: No, you are in a buffer.

How can I quit my `vi` session without saving my changes? Answer: `:q!`.

How many changes will `u` undo? Answer: One; it will only undo the *last* change.

Why does `u` work as a toggle? Answer: `u` can undo itself.

Where Problems Arise

The students need to see the distinction between their files and the buffer.

The `ZZ` command makes no sense! It is just a fast way to save your changes and leave `vi`. Shift zz is a quick combination of keystrokes.

Evaluation Questions

What would happen if we did not save the buffer, but just quit `vi`?

Answer: We would get a message at the bottom of the screen saying "No write since last change (:quit! overrides)"

How would we recover if we "trashed" our file while in `vi` and did not want to lose the original text?

Answer: Type `:q!`.

Presentation Suggestions

Use "reveal" to illustrate each command in turn.

Transition

Let's try some of these administrative commands.

7-5. LAB: Administration Exercises

1. Press Ctrl-g and notice the status line at the bottom of your screen.
2. Use the `write` command to save this file as a file called `another.jersey`.
3. Use `:q` to exit `vi`. Why didn't you need to use `:q!`?

7-5. LAB: Administration Exercises

Purpose

To try the new `vi` commands.

Transition

The cursor is the pointer into our file, so moving our cursor moves us around the file.

Solutions

1. Press Ctrl-g and notice the status line at the bottom of your screen.
2. Use the `write` command to save this file as a file called `another.jersey`.

Answer:

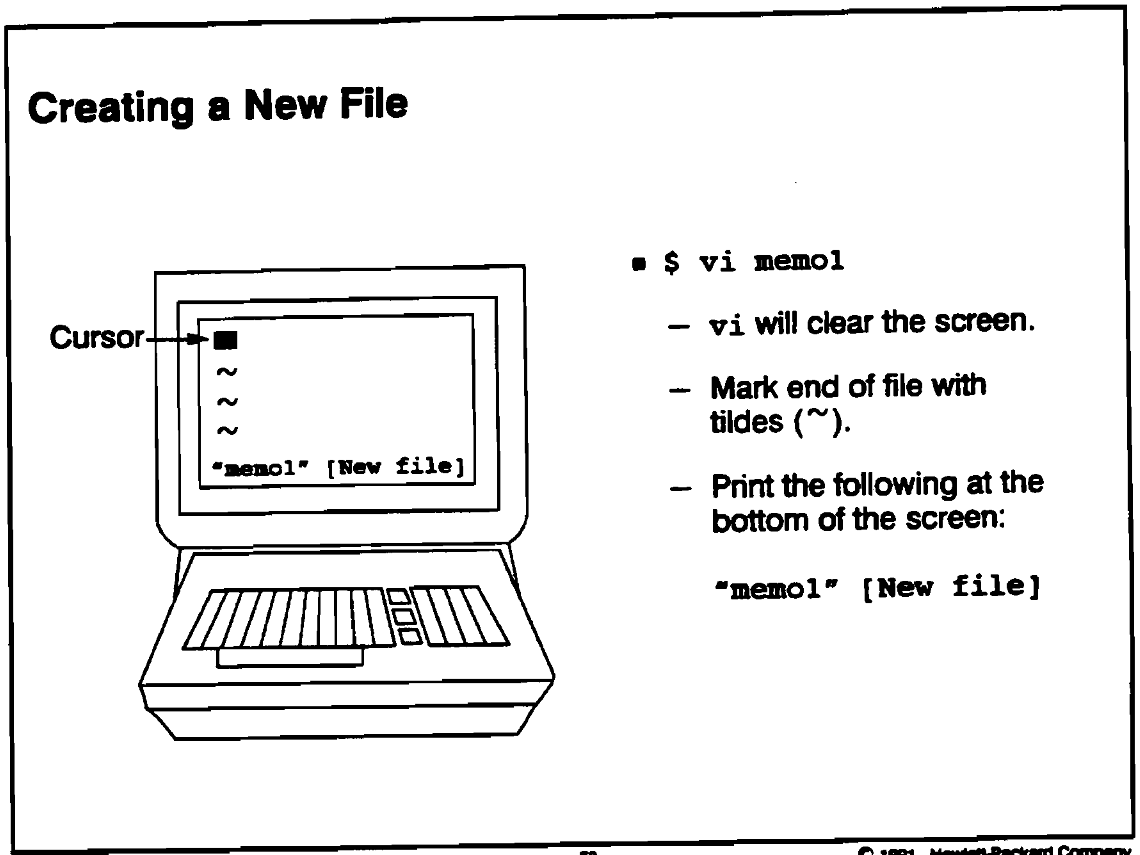
While in `vi` type `:w another.jersey`.

3. Use `:q` to exit `vi`. Why didn't you need to use `:q!`?

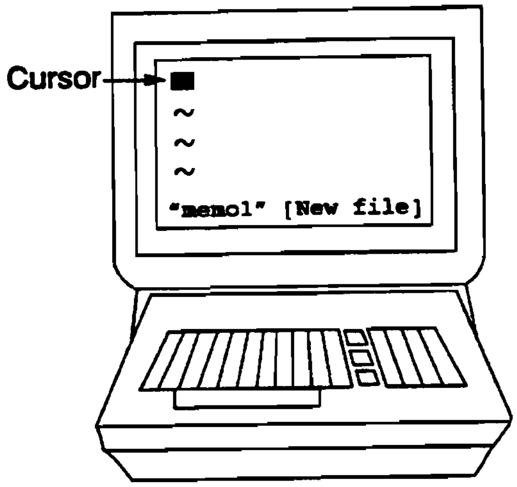
Answer:

You did not make any changes to the file, so `:q` let you quit.

7-6. SLIDE: Creating a New File



Creating a New File



- `$ vi memol`
  - `vi` will clear the screen.
  - Mark end of file with tildes (~).
  - Print the following at the bottom of the screen:

    `"memol" [New file]`

51489 7-6. 83 © 1991 Hewlett-Packard Company

Student Notes

If we wanted to create a new file with a memo to our staff called "memol," we would type in the following:

```
$ vi memol
```

`vi` will clear your screen and place tilde characters ("~ ") down the left margin to mark the end of the file. It will also give you the message:

```
"memol" [New file]
```

on the bottom left of your screen. Your cursor will be in the upper left-hand corner of the screen.

At this point, you are in command mode and `vi` will wait for you to tell it what to do.

Exercises

1. Using `vi`, edit a new file called `memo1`. Notice what creating a new file looks like.
2. Exit `vi` using ZZ. Did a file get created in your directory?

7-6. SLIDE: Creating a New File

Purpose

To show how to create a new file and what a new file looks like in `vi`.

Key Points

This shows how we can create a new file using `vi`. The syntax is the same as for getting a file out of storage. The slide shows what the screen looks like when we create a new file.

Teaching Question

Are the tildes you see part of your file? If not, what is their purpose?

Transition

The last point on this slide is a `vi` message. `vi` has many such messages that may need a little explanation. They are on the next slide.

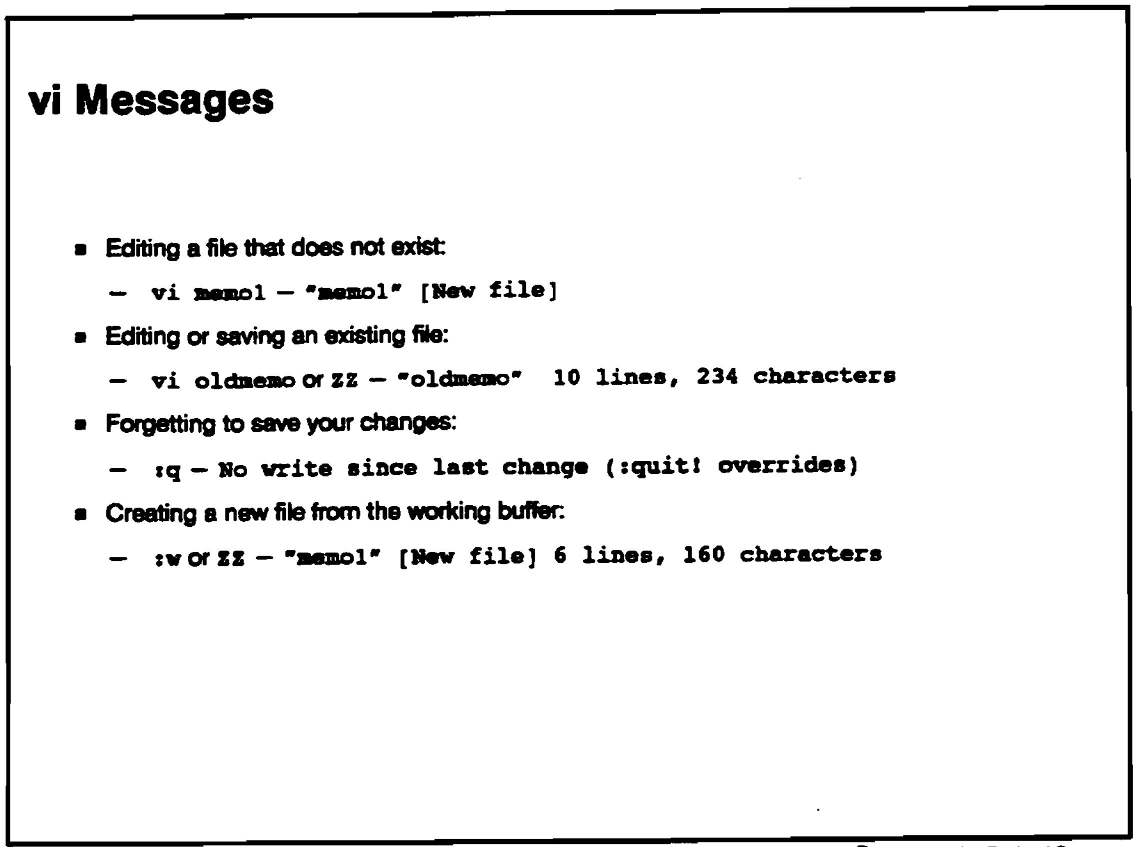
Answers

1. Using `vi`, edit a new file called `memo1`. Notice what creating a new file looks like.
2. Exit `vi` using ZZ. Did a file get created in your directory?

Answer:

NO. `vi` does not store a *new* buffer that is empty.

7-7. SLIDE: vi Messages



vi Messages

- Editing a file that does not exist:
  - `vi memo1` — `"memo1" [New file]`
- Editing or saving an existing file:
  - `vi oldmemo` or `ZZ` — `"oldmemo" 10 lines, 234 characters`
- Forgetting to save your changes:
  - `:q` — `No write since last change (:quit! overrides)`
- Creating a new file from the working buffer:
  - `:w` or `ZZ` — `"memo1" [New file] 6 lines, 160 characters`

51489 7-7. 54 © 1991 Hewlett-Packard Company

Student Notes

`vi` has its own set of commands, prompts, and status messages. However, `vi` is fairly quiet; it does not give you messages very often. There are a few printed messages it might give you that will appear at the bottom of the screen. You should understand what each means:

- `"memo1" [New file]`

  This message means that you invoked `vi` on `memo1` and it does not exist in the current directory.

- `"oldmemo" 10 lines, 234 characters`

  You will get this message when you invoke `vi` on a file that exists, and when you write the file using either ZZ or `:w`.

- `No write since last change (:quit! overrides)`

This is a reminder to save the changes you have made to a file before you quite `vi`. If you really want to leave `vi` without saving your changes, you can use `:q!`; otherwise, make sure your changes are saved using `:w` or ZZ.

- `"memo1" [New file] 6 lines, 160 characters`

  This verifies that `memo1` has been created and stored in the current directory.

When you do something that `vi` does not understand, `vi` will make your computer terminal beep at you. This is one of the frustrations when you are first learning `vi`.

7-7. SLIDE: vi Messages

Instructor Notes

Purpose

To explain the messages that `vi` prints at the bottom of the screen.

Key Points

`vi` has various messages that appear at the bottom of the screen. Otherwise, `vi` is silent.

Teaching Question

Where do *all* `vi` messages appear?

Answer: At the bottom of the screen.

Evaluation Questions

If no messages have appeared at the bottom of the screen and your terminal has not beeped, what should be true of the last command you typed? Either the bell on your terminal is not working or you entered a command that was correct!

Presentation Suggestions

Use "reveal" on the slide.

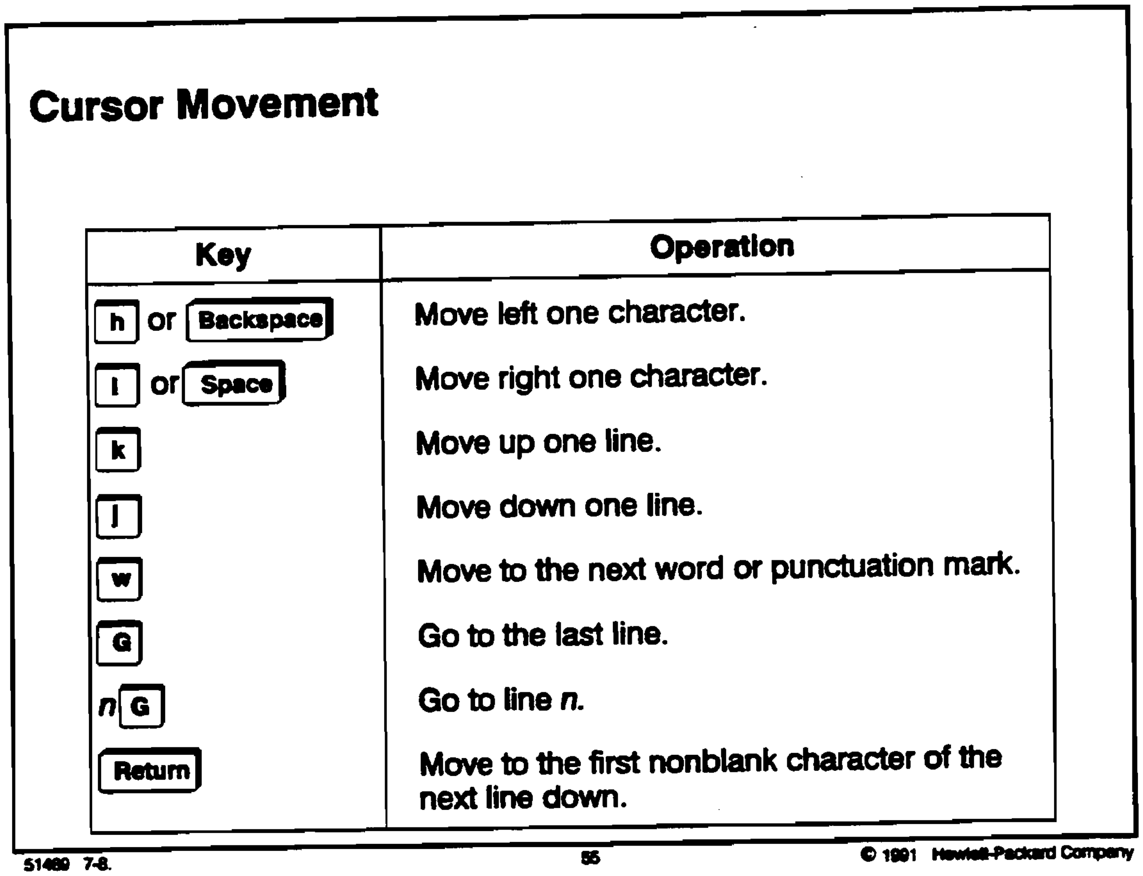
Slide Notes

Explain each message in turn.

Transition

Let's practice moving around in our files in `vi`

7-8. SLIDE: Cursor Movement



Cursor Movement

Key	Operation
h or Backspace	Move left one character.
l or Space	Move right one character.
k	Move up one line.
j	Move down one line.
w	Move to the next word or punctuation mark.
G	Go to the last line.
n G	Go to line *n*.
Return	Move to the first nonblank character of the next line down.

51489 7-8. 55 © 1991 Hewlett-Packard Company

Student Notes

To change the window into our file, we need to move around in our file. We do this with `vi`'s cursor movement commands. You can only move around your file when you are in command mode.

Basic Moving Keys

The basic movement keys move the cursor right and left over existing text in a file. The cursor will not move past the beginning or end of a line. (`vi` will beep if you try).

`h` and backspace	Move the cursor left one character.
`l` (ell) and space	Move the cursor right one character.

There are also cursor movement keys to move the cursor up and down a line. You can use these keys to go from the end of the file toward the beginning, or from the beginning of the file toward the end. `vi` will beep if you try to move past the beginning or end of the file. These keys are the following:

j	Move the cursor down one line, in the same column if possible.
k	Move the cursor up one line, in the same column if possible.

We use h, j, k, and l as our movement keys because they are right next to the home typing keys (j, k l, and ;). This makes it very easy to jump between the typing keys and movement keys without having to search the keyboard for special movement keys.

If your terminal has keys with arrows on them, they may also move the cursor one position in the direction that the arrow points. However, the arrow keys do not work on all types of terminals.

Moving by Words

Moving one character at a time is handy, but there are times when we want to move more quickly within a file. We can do this with w. This command moves the cursor forward by words.

w	Advance the cursor to the beginning of the next word, stopping at punctuation marks. Notice that a word includes its trailing space.

w will wrap around from the end of one line to the beginning of the next line.

Moving by Lines

If you want to jump several lines at a time or quickly move to the end of the file, you can use the G (goto) command. This command has two commonly used forms:

*n*G	Move the cursor to the first nonblank character on line *n*.
G	Move the cursor to the first nonblank character of the last line

To go to line number 5, type 5G. Do not type Return after the G , because it is not necessary. It will take you to the next line if you do.

Return moves the cursor to the first nonblank character on the next line. If the line is empty, the cursor will go to the end of the line. It is actually on an invisible, nonblank character that represents the end of the line.

Nonblank characters are anything except a space or a tab. It gets more complicated when the line contains only spaces or tabs. Return takes you to the end of the line, which may be in the middle of the screen!

Remember, you can use Ctrl-g to find out which line you are on.

7-8. SLIDE: Cursor Movement

Instructor Notes

Purpose

Since `vi` is our window into the file we are editing, we have to move around the file so we can operate on different parts of it.

Key Points

`vi` is meant to be very fast and efficient to use. All of the commands are within the normal typing keys. This means you will never have to search the keyboard for special keys (unless you are a"hunt and peck" typist).

Nonblank characters are anything except a space or a tab.

Teaching Question

If the cursor is the pointer into the file, then we need a mechanism for moving around in our window. Why do you think h, l, k, and j were chosen as the primary movement keys? The students should realize that these are next to the keyboard home keys.

Where does `G` move the cursor? Where does `1G` move the cursor?

Answer: `G` moves to the last line of the file; `1G` moves to the first line of the file.

If this were an English class, how would you define a word? You might say that it is a group of characters that is separated by white space. That is `W`'s interpretation. Our you may say that punctuation marks are not part of the word. This is how `w` recognizes words.

Where Problems Arise

One place where a problem can arise is with "blank" lines. You cannot tell the difference between lines that contain spaces and tabs and lines that contain nothing at all. `vi` and Return will behave differently for these two conditions even though they appear the same on the screen The `:set list` command can be used to show where lines really end.

Make certain that the definition for `w` is clear for future reference because the students must use it in the more advanced commands.

Evaluation Questions

Will the cursor move off of the current line with h or l?

Answer: No

Where does the cursor move (which column) when going from line to line?

Answer: The cursor goes to the same column, if possible.

After you have invoked vi, what is the quickest way to get to the end of your file?

Answer: G.

Will moving by words wrap around to a new line? Why?

Answer: Yes.

After the exercises ask: Why does the cursor sometimes appear at the left column of a blank line and sometimes appear indented on a blank line?

Answer: Sometimes a blank line contains spaces and tabs and sometimes it contains nothing at all.

Presentation Suggestions

Use the slide to cover one topic at a time.

Chalkboard Notes

Draw a screen on the board with some scribbles that stand for lines of text. Make one or two lines a few spaces off of the left margin. Ask what will happen if you press Return and move the cursor down the screen until you get to the last line. Ask what happens when you are on the last line of the screen and you press Return.

Transition

Now that we know how vi acts, we will want to know how we can insert text.

7-9. LAB: Moving Exercises

1. Use `vi` to edit the file called `numbers` and follow the instructions there. Also try moving down the lines by pressing Return. Why does Return take you to the right on some of the blank lines?

When you are done, exit the file using `:q!`.

2. Edit the file called `new.jersey` and move to the last line.

Then, move to line 30.

Then, move to line 1.

How can you tell that you are on the correct line?

3. Move to the beginning of line 26 in `new.jersey` and type `w` five times.

4. Exit the file with `:q!`.

5. Advanced: `w` moves you forward by words. How can you move backwards by words?

7-9. LAB: Moving Exercises

Purpose

To gain some practice moving around files before we go on to the other `vi` functions.

Transition

Let's see how to create a new file.

Solutions

1. Use `vi` to edit the file called `numbers` and follow the instructions there. Also try moving down the lines by pressing Return. Why does Return take you to the right on some of the blank lines?

When you are done, exit the file using `:q!`.

Answer:

Return takes you to the first nonblank character of the next line. The blank lines near the top of the file are truly empty, so the first nonblank character is at the beginning of the line. Near the bottom of the file, the lines are filled with spaces, so the first nonblank character is at the end of the blanks.

2. Edit the file called `new.jersey` and move to the last line.

Then, move to line 30.

Then, move to line 1.

How can you tell that you are on the correct line?

Answer:

`G` moves you to the last line, `30G` moves you to line 30, and `1G` moves you to line 1. In each case, you can use Ctrl-g to verify that you are on the correct line.

3. Move to the beginning of line 26 in `new.jersey` and type `w` five times.

Answer:

`w` stops at the punctuation marks.

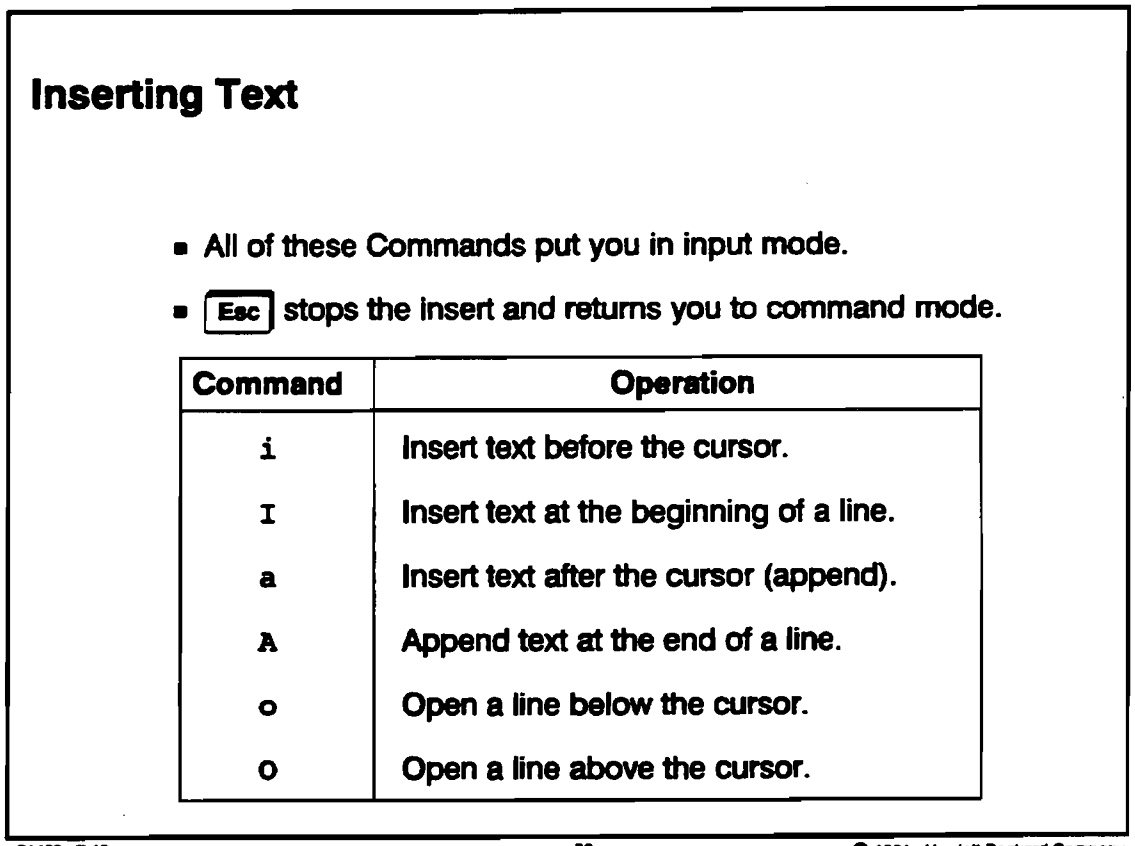
4. Exit the file with `:q!`.

5. Advanced: `w` moves you forward by words. How can you move backwards by words?

Answer:

The b command works just like the w command, only backwards.

7-10. SLIDE: Inserting Text



Inserting Text

- All of these Commands put you in input mode.
- Esc stops the insert and returns you to command mode.

Command	Operation
`i`	Insert text before the cursor.
`I`	Insert text at the beginning of a line.
`a`	Insert text after the cursor (append).
`A`	Append text at the end of a line.
`o`	Open a line below the cursor.
`O`	Open a line above the cursor.

51489 7-10. 56

Student Notes

When you insert text, you are entering `vi`'s "input mode." There are several commands that allow you to insert text. The slide shows these commands. As with all `vi` commands, pay attention to the case of the letters (uppercase or lowercase).

Once you are in input mode, the Esc key will return you to command mode. The Esc key ends any command that adds text. Remember, if you are not sure which mode you are in, you can press Esc and you will be in command mode. If you are in command mode already, vi will beep at you.

If you make an error while entering text, you can correct it by backing up over the error using the backspace key and retyping the text. The backspace key backs up one character at a time on the current line. Notice that it does not erase the the characters, but it allows you to type over them.

Ctrl-x or @ (the kill character) deletes all characters entered on the current line. It backspaces over everything added since you last entered input mode (the same as pressing the backspace key many times). Ctrl-x or @ is the standard kill character. If it does not work on your terminal, ask your system administrator what the the kill character is.

7-10. SLIDE: Inserting Text

Instructor Notes

Purpose

These are the `vi` commands that allow you to insert text. The Esc key will take you out of text input mode.

Key Points

`i`	Allows you to insert text *before* the character the cursor is on.
`I`	Allows you to insert text at the *beginning* of the line the cursor is on.
`a`	Allows you to append text *after* the character the cursor is on.
`A`	Allows you to append text at the *end* of the line the cursor is on.
`o`	Opens the line *below* the cursor for input.
`O`	Opens the line *above* the cursor for input.

When you want to stop inserting text, press Esc. You will return to the command mode.

The backspace key and Ctrl-x or @ (the kill character) will allow you to back up over characters while you are still in input mode. It is important to do `stty -a` to determine which characters will work on your terminal. We will see how to get rid of unwanted characters while you are in command mode shortly.

Presentation Suggestions

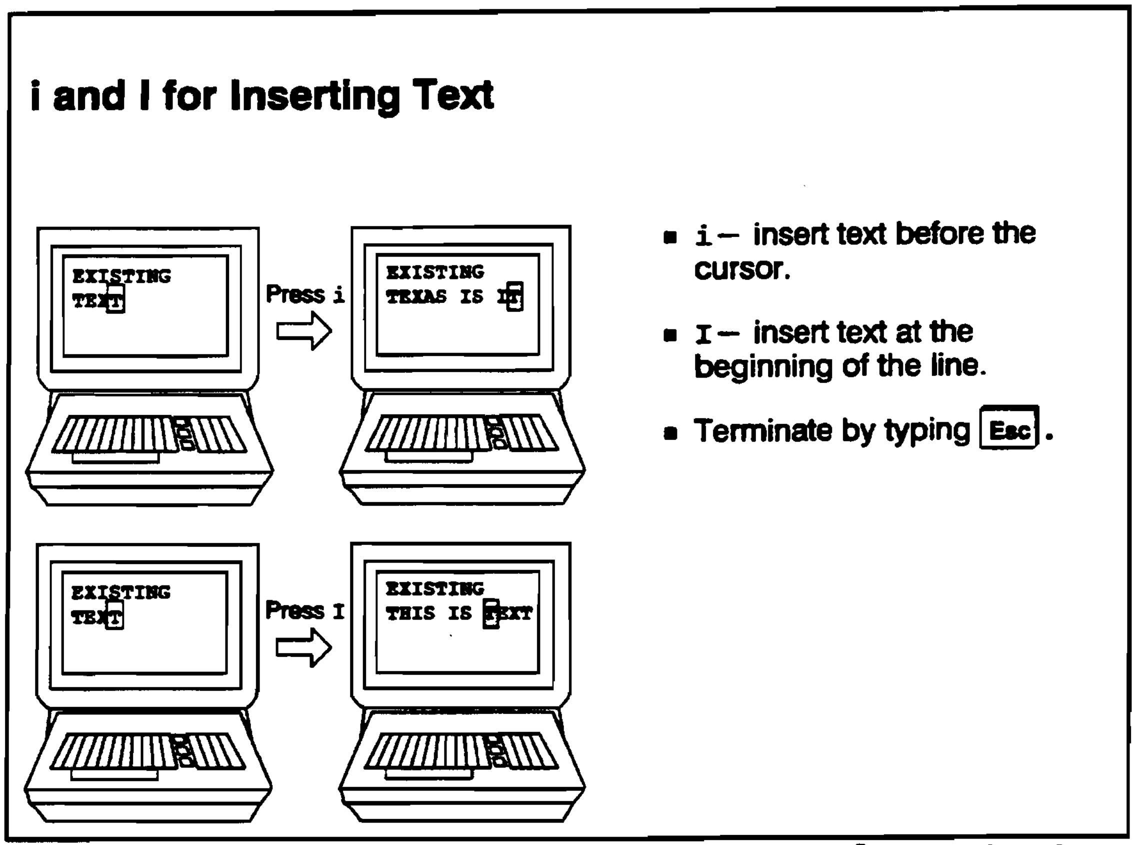
Move through the next few slides *quickly.* The material is not difficult, and students will get more out of the exercises.

Use "reveal" to highlight each command.

Transition

Let's look at each of these commands in detail. We'll look at the `i` (insert) command first.

7-11. SLIDE: i and I for Inserting Text



i and I for Inserting Text

EXISTING
TEXT
Press i
EXISTING
TEXAS IS IT
EXISTING
TEXT
Press I
EXISTING
THIS IS TEXT

- `i` – insert text before the cursor.
- `I` – insert text at the beginning of the line.
- Terminate by typing Esc.

51489 7-11. 57 © 1991 Hewlett-Packard Company

Student Notes

`i` stands for "insert" text.

Typing `i` will put you in input mode immediately *before* the current cursor position. Then you can insert as much text as you want (one character or many lines) until you press Esc. Remember that the `i` command will not be displayed on your screen.

If you use `I` instead, `vi` will put you in input mode at the *beginning* of the current line.

When you have finished inserting text, press Esc. `vi` will leave text input mode and return you to command mode.

7-11. SLIDE: i and I for Inserting Text

Instructor Notes

Purpose

The ability to append text before the cursor.

Key Points

`i` will place you in input mode immediately before the cursor. `I` will place you in input mode at the beginning of the current line.

Teaching Question

How can you insert text at the beginning of a line?

Answer: `I`.

Evaluation Questions

What is the difference between `i` and `I`?

Answer: `I` inserts text at the *beginning* of a line; `i` inserts text right before the cursor position.

Why would we want both? Do we need both?

Answer: Although we do not *need* both, `I` lets us move to the beginning of a line quickly.

Slide Notes

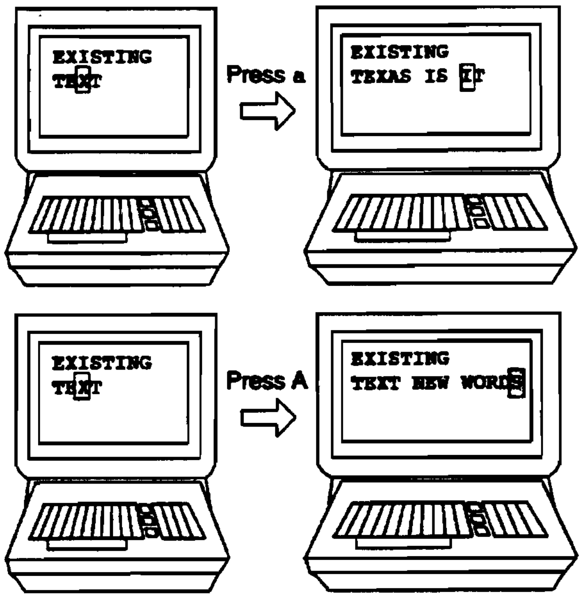
Emphasize that Esc is necessary to end `i` and `I`.

Transition

How can we add text to the end of a line using `i`? It would not be easy, so we have the a (append) command.

7-12. SLIDE: a and A for Appending Text

a and A for Appending Text



- a – append text after the cursor.
- A – append text at the end of the line.
- Terminate by typing Esc.

51489 7-12. 58 © 1991 Hewlett-Packard Company

Student Notes

a stands for "append."

Typing a will put you in input mode immediately *after* the current cursor position. Every character you type will be appended to the buffer until you type Esc.

Typing A will put you in input mode at the *end* of the current line.

7-12. SLIDE: a and A for Appending Text Instructor Notes

Purpose

Since just having `i` would be very inconvenient, we have a way to add text after the cursor.

Key Points

`a` will put us in input mode immediately after the cursor. `A` will put us in input mode at the end of the current line.

Teaching Question

How much text can I add after typing an `a`? How do I stop appending text?

Answer: You may add as much text as you wish until you type Esc.

Where Problems Arise

This is straightforward, but students often forget to type Esc to end the input mode.

Evaluation Questions

Can I append a carriage return to break a line into two lines?

Answer: Yes

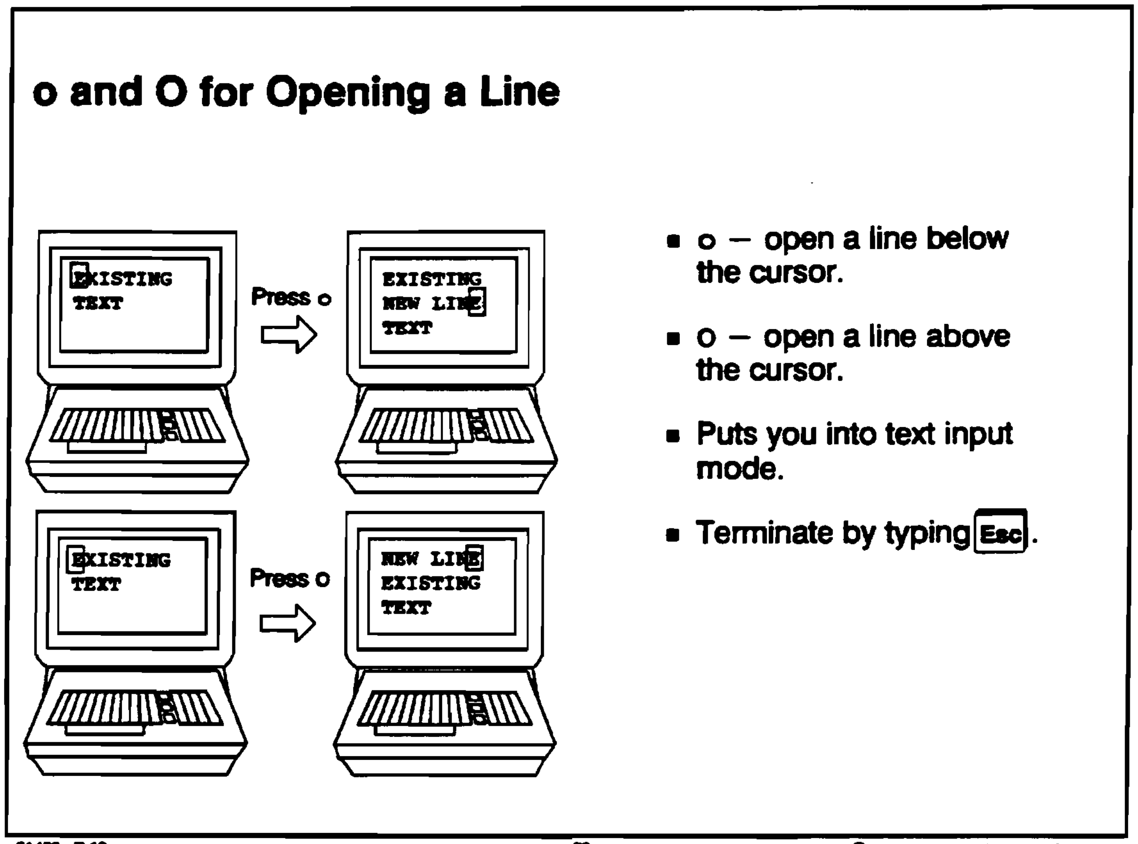
Does input mode look any different if we enter it with `i` or `a`?

Answer: No.

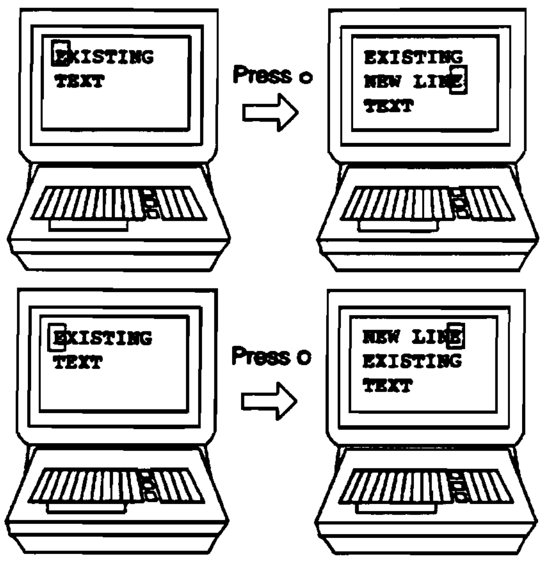
Transition

What if we wanted to enter text just above or just below the current line?

7-13. SLIDE: o and O for Opening a Line



o and O for Opening a Line



- o – open a line below the cursor.
- O – open a line above the cursor.
- Puts you into text input mode.
- Terminate by typing Esc.

51489 7-13. 59 © 1991 Hewlett-Packard Company

Student Notes

o and O allow you to "open" a new line in your buffer.

o will open a line *below* the cursor and put you in input mode until you type an Esc.

O will open a line *above* the cursor and put you in input mode until you type an Esc.

When an o or O is typed, a blank line opens and you are ready to type as much text as you like. When you have finished, press Esc and vi will return you to command mode.

7-13. SLIDE: o and O for Opening a Line

Instructor Notes

Purpose

To open lines with `i` or `a` would mean moving to the beginning or end of the line first. `o` makes opening lines easier.

Key Points

`o` opens a line below the cursor and `O` opens a line above the cursor.

Teaching Question

How can you enter text before the first line of your file?

Answer: `O`.

Where Problems Arise

This is straightforward, however, emphasize that Esc is necessary to end the text input mode.

Evaluation Questions

Can you do the same thing that `o` does using `i` or `a`?

Answer: Yes, use `a` and Return.

Can you do the same thing that `O` does using `i` or `a`?

Answer: Yes, use `i` and Return.

Presentation Suggestions

Go over each example on the slide picture.

Slide Notes

Explain that they both need an Esc.

Transition

We have now seen the basic commands `vi` has for entering the input mode. Let's try some exercises using these new commands.

7-14. LAB: Inserting Text

1. Now, using `vi`, edit a new file called append, type an a, and then enter the following:

```
Note that typing an a while in command mode does not produce output,
but puts you into text input mode.  Now if you type commands in input mode:

aaaaaaaaaaa w w w w ZZ ZZ ZZ

they are just appended to the buffer.
```

Press (Esc). Now save the contents of the buffer and leave vi.

2. Using `vi`, edit a file called `add.text`. Move to the end of the first line, type an a, a carriage ((Return)), and enter this line:

```
This line is being appended to my buffer using "a".
```

Then press (Esc) to leave input mode. Where did the text get appended in relation to the cursor?

3. Now, move the cursor to the first line and type an I. Add the following line:

```
This line is being inserted into the buffer using "I".
```

Type a carriage return and an (Esc). Where does text get inserted in relation to the cursor? Save the contents of the buffer and leave `vi`.

4. Could you have accomplished these same tasks using o and O?

5. Using `vi`, edit the file you created called append. Using only *two* commands, go to the last line and add the words `new test` to the end of the line. *Do not press* (Esc) *yet.*

Where was the text added?

6. While you are still in input mode, use the backspace key to back up over the word `test` and type `text`. What happened?

Press Esc to leave input mode. Save your changes and exit `vi`.

7. Using `vi`, edit the file called `add.text` again. Put the cursor on the first line and type an `o` and then this line:

```
This line was opened up in my buffer using "o".
```

Then type an Esc. Where was the text added?

8. Now put the cursor on the first line and type an `O`. Then add this line:

```
This line was opened in my buffer using "O".
```

Then type an Esc. Where was the text added?

9. Could we have done this with an `a`?

10. Type `ZZ` to save the file and exit `vi`.

7-14. LAB: Inserting Text

Instructor Notes

Purpose

To practice the six new ways to insert text.

Transition

Now that we have text, how can we get rid of the text we do not want?

Solutions

1. Now, using `vi`, edit a new file called append, type an `a`, and then enter the following:

```
Note that typing an a while in command mode does not produce output,
but puts you into text input mode.  Now if you type commands in input mode:

aaaaaaaaaaa w w w w ZZ ZZ ZZ

they are just appended to the buffer.
```

Press Esc. Now save the contents of the buffer and leave vi.

2. Using `vi`, edit a file called `add.text`. Move to the end of the first line, type an `a`, a carriage (Return), and enter this line:

```
This line is being appended to my buffer using "a".
```

Then press Esc to leave input mode. Where did the text get appended in relation to the cursor?

Answer:

After the cursor. Return also gave us a new line.

3. Now, move the cursor to the first line and type an `I`. Add the following line:

```
This line is being inserted into the buffer using "I".
```

Type a carriage return and an Esc. Where does text get inserted in relation to the cursor? Save the contents of the buffer and leave `vi`.

Answer:

`I` inserts text at the beginning of the line, regardless of where the cursor is on that line.

4. Could you have accomplished these same tasks using `o` and `O`?

Answer:

Yes! `vi` has several ways to do everything. The more sophisticated commands usually take less work to get the same results.

5. Using `vi`, edit the file you created called `append`. Using only *two* commands, go to the last line and add the words `new test` to the end of the line. *Do not press* Esc *yet.*

Where was the text added?

Answer:

`G` takes you to the last line and `A` puts you in input mode at the end of that line.

6. While you are still in input mode, use the backspace key to back up over the word `test` and type `text`. What happened?

Press Esc to leave input mode. Save your changes and exit `vi`.

Answer:

The backspace key backed the cursor up over the word `test` and allowed you to overstrike it with the word `text`. Note that in input mode, the backspace key moves you to the left by one character.

7. Using `vi`, edit the file called `add.text` again. Put the cursor on the first line and type an `o` and then this line:

```
This line was opened up in my buffer using "o".
```

Then type an Esc. Where was the text added?

Answer:

On the line after the cursor.

8. Now put the cursor on the first line and type an `O`. Then add this line:

```
This line was opened in my buffer using "O".
```

Then type an Esc. Where was the text added?

Answer:

A line was opened before the line the cursor was on. The line you typed is now the first line of the file.

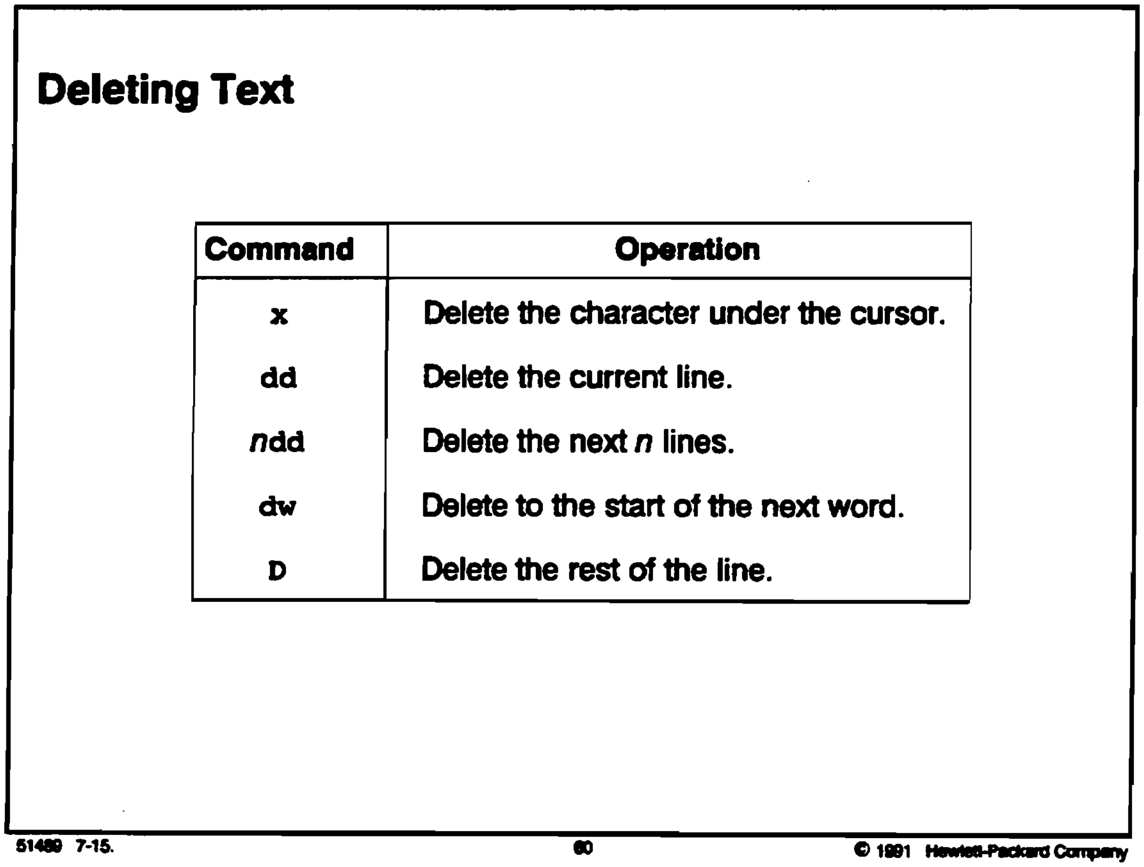
9. Could we have done this with an `a`?

Answer:

No, because we can only add text after the cursor with `a`.

10. Type `ZZ` to save the file and exit `vi`.

7-15. SLIDE: Deleting Text



Deleting Text

Command	Operation
x	Delete the character under the cursor.
dd	Delete the current line.
*n*dd	Delete the next *n* lines.
dw	Delete to the start of the next word.
D	Delete the rest of the line.

51489 7-15. 60 © 1991 Hewlett-Packard Company

Student Notes

When you are editing a file, you may wish to delete some of the text. The commands in the slide allow us to delete a single character or several characters at once.

You must be in *command mode* to use any of these delete commands.

x	Delete the single character under the cursor (cross it out).
dd	Delete the entire current line. The cursor can be anywhere on the line to delete it. All text will be removed, including the carriage return.
*n*dd	Delete *n* lines, starting at the current line.
dw	Delete the rest of the word that the cursor is on, stopping at punctuation. If your cursor is on the beginning of the word, the entire word will be deleted. Trailing white space is also deleted.
D	Delete the character under the cursor and the rest of the line.

`x` is fairly straightforward. `d` is `vi`'s delete command. We can use it to delete any number of things by combining it with a movement command such as `w`. We could also use `dj` to delete the current line and the line below the current line. If we put a number in front of the `d`, `vi` will delete that many items. For example, `3dw` will delete three words, and `5dd` will delete 5 lines.

You can also use a count on commands such as `x` to delete several characters with one command. `7x` will delete seven characters.

The ability to combine a count and an object with any command is one of the most powerful features of `vi`.

On older computer terminals or ones that are running at a very slow speed, `vi` may replace the deleted line with an "@". The line has been removed, and the "@" is not part of the buffer; it is just a place holder until you redraw your screen (using Ctrl-l).

7-15. SLIDE: Deleting Text

Instructor Notes

Purpose

These are the `vi` commands that allow you to delete text. They remove unwanted characters and lines from the buffer.

Key Points

See the list on the page.

Teaching Question

Does the `x` need an escape? No, because it is a command and does not put us into input mode.

Evaluation Questions

What happens if you press `x` five times? Answer: Five characters are deleted.

Does it matter where the cursor is on the line when you use `dd` ? Answer: No.

How do we move by words? How do we delete a word? Answer: We use `w` to move by words; we use `dw` to delete a word.

How would we delete from where the cursor is to the end of our file? We would use `dG`.

Where Problems Arise

This is an example of movement being strung together with another `vi` command. The concept is not difficult, but students may need some examples.

Presentation Suggestions

Use "reveal" to highlight each command.

Transition

It's time to practice deleting text.

7-16. LAB: Deleting Text

1. Using `vi`, edit a new file called oops. Add the following line to the buffer;

```
This sentence contains an inncorrrrectlyy spelled word.
```

Then type an (Esc).

Now, move the cursor to the letters that do not belong and type `x`.

2. Type `ZZ` to save the file and exit `vi`.

3. Using `vi`, edit the file called `delete`. Delete the two blank lines in the middle of the file with one command.

4. Move the cursor to the `T` in `THIS` on the line that says `THIS LINE SHOULD BE DELETED`. Now, type `dd`. What happens?

5. Use `u` to undo the deletion. Move the cursor somewhere else on the line and type `dd`. Do this cycle several times. Does it make a difference where the cursor is on the line?

6. Type `ZZ` to save the file and exit `vi`.

7. Using `vi`, edit a file called `unwanted`. Move to line 31 and put your cursor on the string `WORD`. Type `dw`. What happens?

8. Put your cursor on the string `WORD.WORD` on line 70 and type `dw`.

9. Type `u` and then delete `WORD.WORD`.

10. Type `ZZ` to save the file and exit `vi`.

7-16. LAB: Deleting Text

Instructor Notes

Purpose

To practice using vi's delete commands.

Transition

What if we did not want to delete the text but want to change it to something else?

Solutions

1. Using vi, edit a new file called oops. Add the following line to the buffer;

```
This sentence contains an inncorrrrectlyy spelled word.
```

Then type an Esc.

Now, move the cursor to the letters that do not belong and type x.

Answer:

The letter under the cursor is deleted.

2. Type ZZ to save the file and exit vi.

3. Using vi, edit the file called delete. Delete the two blank lines in the middle of the file with one command.

Answer:

Use 2dd. The text will close up and redraw or will leave behind "@" characters, as you delete the lines.

4. Move the cursor to the T in THIS on the line that says THIS LINE SHOULD BE DELETED. Now, type dd. What happens?

Answer:

The line is deleted.

5. Use u to undo the deletion. Move the cursor somewhere else on the line and type dd. Do this cycle several times. Does it make a difference where the cursor is on the line?

Answer:

dd deletes the whole line regardless of where the cursor is on that line.

6. Type ZZ to save the file and exit vi.

7. Using vi, edit a file called unwanted. Move to line 31 and put your cursor on the string WORD. Type dw. What happens?

Answer:

The word disappears from the text.

8. Put your cursor on the string WORD.WORD on line 70 and type dw.

Answer:

Just the first WORD was deleted because w stopped at the ".".

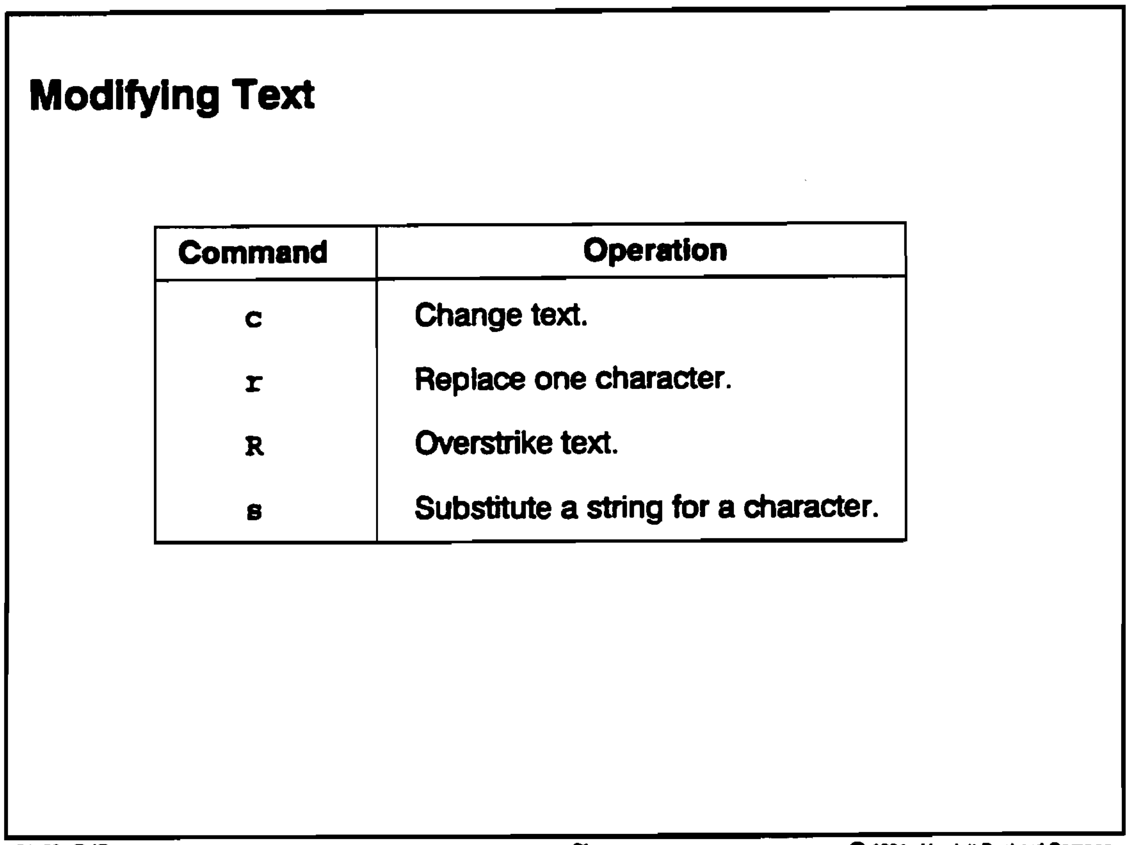
9. Type u and then delete WORD.WORD.

Answer:

3dw would delete WORD.WORD.

10. Type ZZ to save the file and exit vi.

7-17. SLIDE: Modifying Text



Modifying Text

Command	Operation
c	Change text.
r	Replace one character.
R	Overstrike text.
s	Substitute a string for a character.

51489 7-17. 61

Student Notes

This slide presents the commands you need so that you can modify or update a file. You execute these commands while in the command mode.

You already know how to move the cursor to an error, delete it, and then insert the correct text. With the change commands, you can use c (change), r (replace), R (overstrike), or s (substitute) to correct errors in a single step.

7-17. SLIDE: Modifying Text

Instructor Notes

Purpose

These are the `vi` commands that allow you to modify text.

Key Points

`c` is the change command in `vi`.

`R` and `r` are the replace commands in `vi`.

`s` is the substitute command.

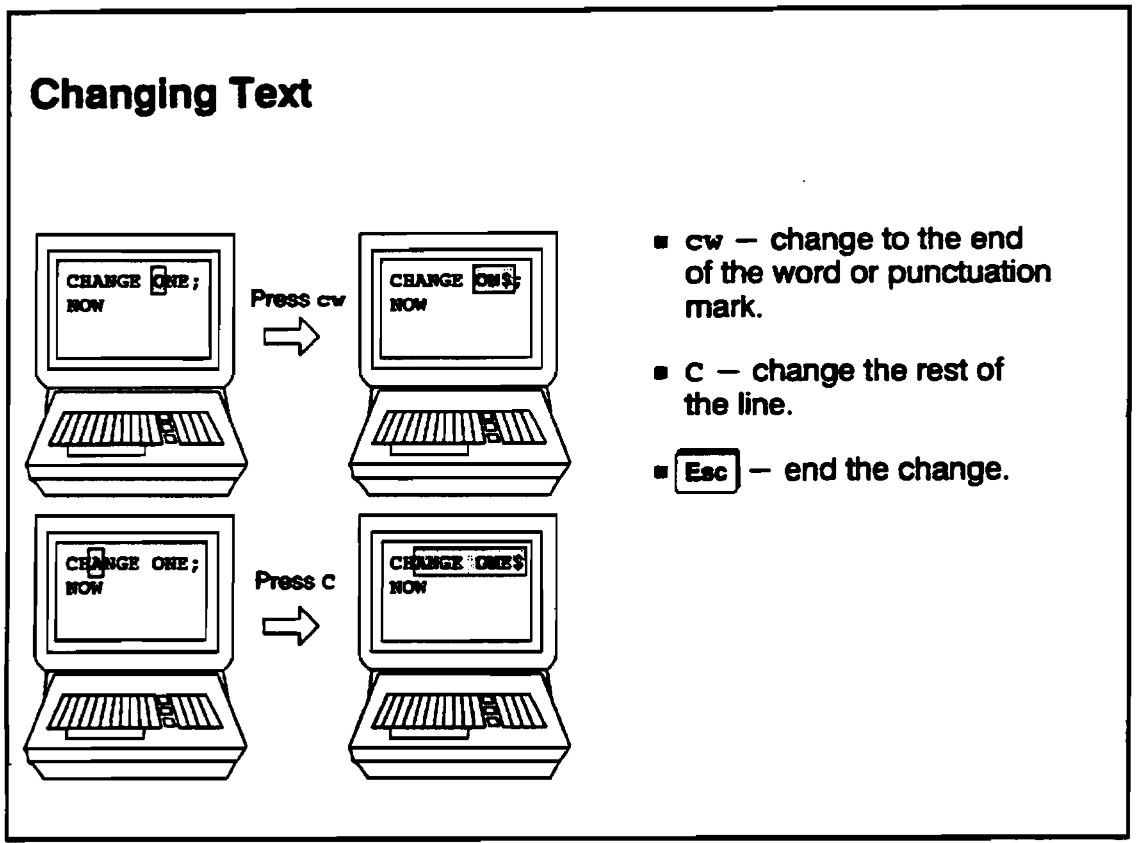
Presentation Suggestions

Use "reveal" to highlight each command.

Transition

Let's look at each of these commands in detail. We'll look at the `c` change commands first.

7-18. SLIDE: Changing Text



Changing Text

- cw – change to the end of the word or punctuation mark.
- c – change the rest of the line.
- Esc – end the change.

51469 7-18. 62 © 1991 Hewlett-Packard Company

Student Notes

Suppose that you do not like a particular word in your file and think that another word would more accurately convey what you want to say. We would like to change the word we have into another word or phrase.

The change command in `vi` is `c`. Combine this with the word movement function, `w`, and you can change any word.

`cw`*text* Esc	Change the word on which the cursor rests (up to the next word or punctuation) to "*text*".
`C`*text* Esc	Change the character the cursor is on and the rest of the line to "*text*".

When you type any of the change commands, a dollar sign ($) appears at the end of what you are changing. This is `vi`'s way of letting you know what it thinks is going to change. The text between the cursor and the dollar sign remains, and you overstrike the old text with what you type. When you enter a change command, you are in input mode. This means that when you are done changing the text, you must press Esc.

cw is an example of combining a command (c) with cursor motion (w). c will work with other cursor motions as well. You can also put a number in front of the c you want changed. For instance, 2cw will change two words.

7-18. SLIDE: Changing Text

Purpose

To understand that c is vi's change command and that when it is combined with cursor movement commands, it can change objects.

Key Points

See the list in the student notes.

Remember that c needs Esc to get you back to command mode.

Teaching Question

How do we move by words? How do we change a word?

Answer: Use w to move by words; use cw to change a word.

Evaluation Questions

How do we change the line we are on and the next line down? We type cj.

Presentation Suggestions

Point out that the ";" is not included in the change on the cw example on the slide.

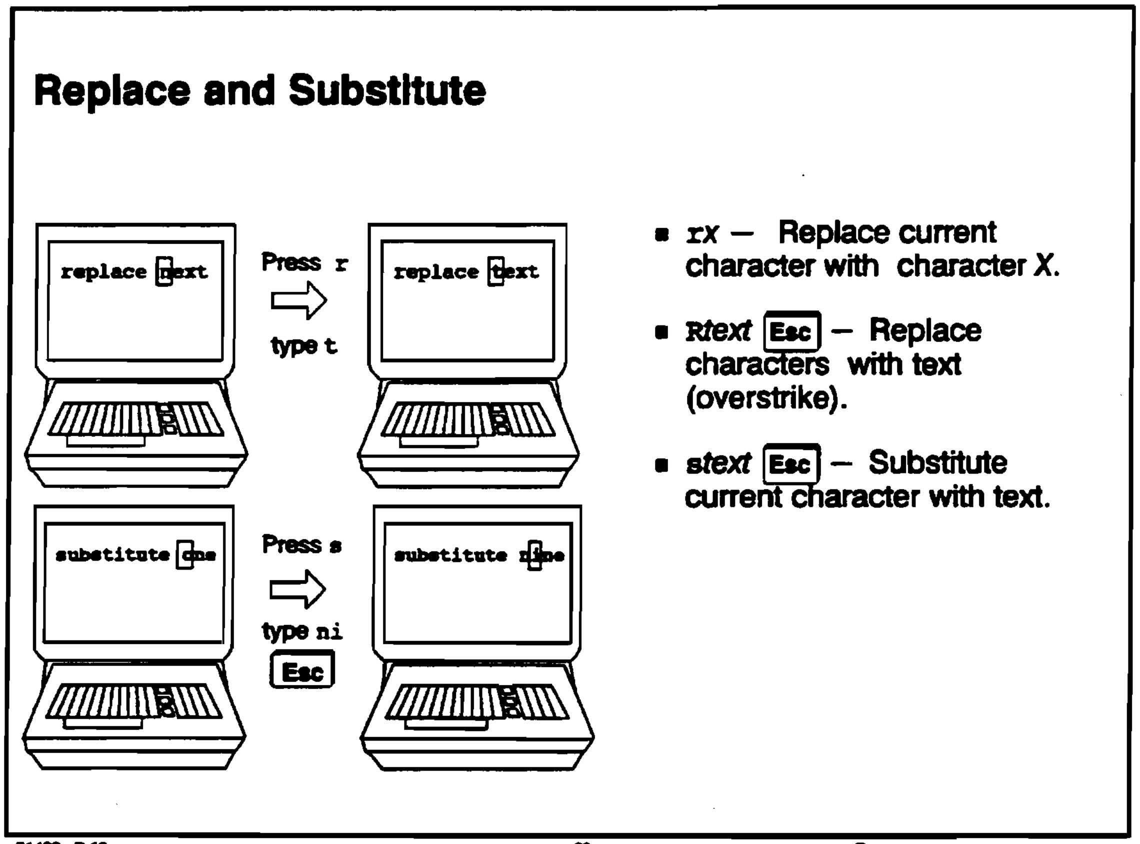
Slide Notes

Make sure that cw is understood first and then move to the concept of a change command.

Transition

Let's take a look at two other ways to modify text in vi.

7-19. SLIDE: Replace and Substitute


replace next
Press r
type t
replace text
substitute one
Press s
type ni
Esc
substitute nine

Replace and Substitute

- *rx* — Replace current character with character *X*.
- *Rtext* Esc — Replace characters with text (overstrike).
- *stext* Esc — Substitute current character with text.

51489 7-19. 63 © 1991 Hewlett-Packard Company

Student Notes

There are two other very useful ways to change text. They are the `r` (replace) and `s` (substitute) commands.

`r`*X*	Replace the character under the cursor with the character *X*.
`R`*text* Esc	Replace each character under the cursor with *text*. This is called overstrike mode.
`s`*text* Esc	Substitute the one character under the cursor with *text*.

`r` and `s` only modify the character that the cursor is on when you type the command. `r` allows you to change just that character to another character, while `s` allows you to change that character to any character, word, or phrase. Notice that `s` requires Esc because it puts you in input mode to make the change.

`R` puts you in overstrike mode. Starting with the character the cursor is consecutive character to the one you type. It also requires an Esc to bring you back to command mode.

7-19. SLIDE: Replace and Substitute

Instructor Notes

Purpose

To introduce the other two most commonly used modification commands.

Key Points

See the list.

Evaluation Question

Why do R and s require an Esc? Answer: Because these commands put you in input mode.

Why doesn't r require an Esc? It only changes one character.

How many characters will r let you change? Only one.

How many characters will R let you change? It will change the rest of a line until you press Esc. If you press Return while in overstrike mode, you will get a new line to continue your modification.

Where Problems Arise

Students sometimes have a hard time with the fact that R and s need Esc but r does not. Otherwise this topic is pretty straightforward.

Chalkboard Notes

A few examples on the board usually help a great deal.

Transition

Let's try it!

7-20. LAB: Modifying Text

1. Using `vi`, edit a file called `mutant`. Find the word `ghouls`, and change it to `food`.

2. Now change the word `food` to `fools` using the `s` command.

3. Change the word `they` to `this` using the `R` command.

4. Use a replace command to change the word `rest` to `best`.

5. Type `ZZ` to save the file and exit `vi`.

7-20. LAB: Modifying Text

Purpose

To practice the three new change commands.

Transition

Let's see how we can copy and move text in a file.

Solutions

1. Using `vi`, edit a file called `mutant`. Find the word `ghouls`, and change it to `food`.

Answer:

Using `cw`, the word `ghouls` can be changed to `food`.

2. Now change the word `food` to `fools` using the `s` command.

Answer:

Move the cursor to the `d` and press `s`. Type `ls` and press Esc.

3. Change the word `they` to `this` using the `R` command.

Answer:

Move the cursor to the `e` and type `R`. Type `is` and press Esc. Notice how each character you typed struck over the character under it.

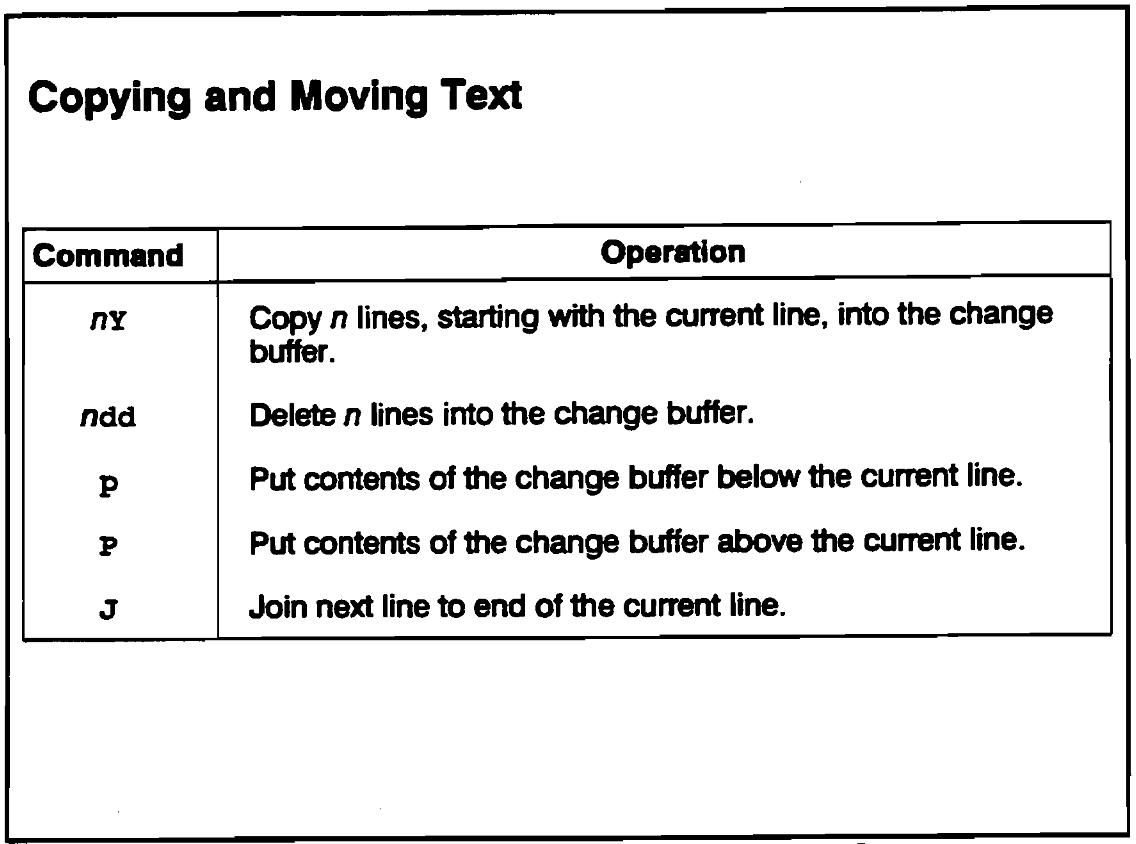
4. Use a replace command to change the word `rest` to `best`.

Answer:

Move your cursor to the `r` in `rest`, press `r` and then `b`, and the change is made.

5. Type `ZZ` to save the file and exit `vi`.

7-21. SLIDE: Copying and Moving Text



Copying and Moving Text

Command	Operation
*n*Y	Copy *n* lines, starting with the current line, into the change buffer.
*n*dd	Delete *n* lines into the change buffer.
p	Put contents of the change buffer below the current line.
P	Put contents of the change buffer above the current line.
J	Join next line to end of the current line.

51480 7-21. 64 © 1991 Hewlett-Packard Company

Student Notes

This slide presents the commands you need so that you can copy and move text commands while in the command mode.

The Y command "picks up" (or "yanks") a copy of the current line into a special buffer called the **change buffer**. This command is used in conjunction with the p or P commands to *copy* text. If you put a count in front of the yank command, you can *yank* several lines, starting with the current line. For example, 5Y will yank the current line and the next four into the change buffer.

The p command "puts" text from the change buffer back into your file right *below* the current line. P puts the change buffer back *above* the current line (the current line is moved down). The put commands will put back all of the lines in the change buffer regardless of how many you have yanked.

To copy six lines of text, you can use 6Y to "pick up" a copy of the lines. Then you use the cursor movement keys to move to another location in the file. When you press p or P, a copy of the lines you yanked will be placed in your file.

To *move* text in your file, you simply have to delete the text to be moved with one of the `d` commands, move to the new location for the text, and use a `p` command to put the text down. All of the delete and change commands use the special change buffer.

Other times we may just want to move a partial line up to the current line to make one full line that looks better. `vi` has the `J` (join) command to do this. Simply place your cursor on a line and type `J`. The line below the current line will be joined onto the end of the current line.

7-21. SLIDE: Copying and Moving Text

Instructor Notes

Purpose

These `vi` command sequences allow you to copy and move text within a file.

Key Points

`Y` is the command used to "pick up" a copy of the current line into the change buffer. You will not see any change on the screen. This command is used in conjunction with the `p` or `P` commands to copy text.

Using a `d` command with `p` allows us to move text.

`J` is `vi`'s command to join two lines.

Presentation Suggestions

Use "reveal" to highlight each command.

Transition

Here are some exercises for you to try copying and moving text.

7-22. LAB: Copying and Moving Text

1. Edit the file called `productivity`. Copy the title line (line 1) and place it at the end of the file so that the statement makes sense.

2. The last two lines are both short. Join the last line with the one above it to make the file look better. Save the corrected file and exit `vi` when you are done.

3. Edit the file called `upside_down`. Use the technique we just learned to move the lines into their correct positions. Save the file and exit `vi`.

4. Advanced:

Using join and replace, make the lines of the file `upside_down` fit the margins of your screen.

7-22. LAB: Copying and Moving Text

Purpose

To practice using the yank and put commands to copy text and the delete and put commands to move text.

Transition

Now let's see how we can search for text in a file.

Solutions

1. Edit the file called `productivity`. Copy the title line (line 1) and place it at the end of the file so that the statement makes sense.

Answer:

On line 1, type `Y` to yank the text. Move to line 4 and type `p` to put the copy of the line down. Moving to the last line and typing `P` would also have worked.

2. The last two lines are both short. Join the last line with the one above it to make the file look better. Save the corrected file and exit `vi` when you are done.

3. Edit the file called `upside_down`. Use the technique we just learned to move the lines into their correct positions. Save the file and exit `vi`.

Answer:

For each line, you will have to delete it using `dd` and then put it where it belongs with `p` or `P`.

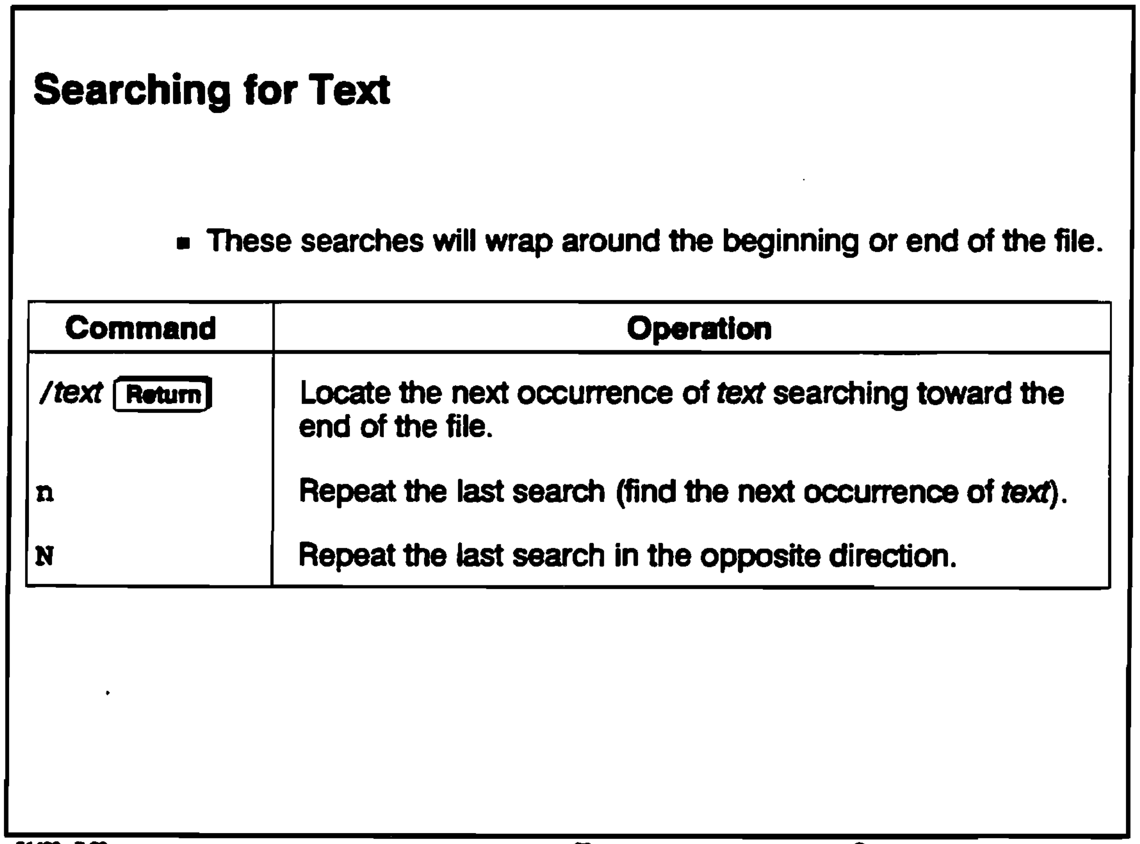
4. Advanced:

Using join and replace, make the lines of the file `upside_down` fit the margins of your screen.

Answer:

Join lines until they wrap around the right margin. Then you can replace a space near the end of the line with Return to split the line into two lines.

7-23. SLIDE: Searching for Text



Searching for Text

- These searches will wrap around the beginning or end of the file.

Command	Operation
/text Return	Locate the next occurrence of *text* searching toward the end of the file.
n	Repeat the last search (find the next occurrence of *text*).
N	Repeat the last search in the opposite direction.

51489 7-23. 65 © 1991 Hewlett-Packard Company

Student Notes

There are times when you will want to find a particular string of characters in your file. Suppose you notice that you typed a word incorrectly or that you have to make an addition to your file after a particular phrase. `vi` hs a mechanism for finding substrings.

The search functions are as follows:

/*text* Return	Search forward through the file for *text*.
`n`	Repeat the last search (next occurrence).
`N`	Repeat the last search, looking in the opposite direction.

Suppose we want to add a paragraph in our file after the paragraph that ends with the words `other lists`. We would `vi` our file and the cursor would be on the first line. We would type the following:

```
/other lists Return
```

This line will appear at the bottom of your screen along with the cursor. When you press Return, `vi` searches forward through the file for the string you typed between the "/" and the Return. If it finds the string, your cursor will be placed on the first character of that string in your text (the `o`). If `vi` could not find the string, the error `Pattern not found` will be displayed on the bottom of your screen, and you will not have moved.

We could type an `n` to find the next occurrence of the string. If we keep typing `n`, we will keep moving to the next occurrence. If `vi` reaches the bottom of the file, it will wrap around to the top and continue searching.

If we passed the occurrence of the string that we wanted to see, we could type an `N`, which looks in the opposite direction (that is, if we were searching forward, it would then search backward).

If you are not sure if the string is in uppercase or lowercase, you can type `:set ic` (ignore case) before you type the search command. To turn off this ignore case option, use `:set noic`.

7-23. SLIDE: Searching for Text

Instructor Notes

Purpose

Find a particular string of text in a file.

Key Points

"/" is used to find strings in `vi`; `n` will find the next instance. All of these functions will wrap.

Use `:set ic` to set the search to be non-case sensitive. Use `:set noic` to reset it.

Teaching Question

How do you look for the word `the`? Note that doing `/the` and `/ the` are two very different things.

Where Problems Arise

Some students may not know what "string" means. Explain that a string is any group of characters. It can be a few characters, a word, or a whole phrase.

Evaluation Questions

By using "/" and `n`, will I find every occurrence of a particular string in a file?

Answer: Yes, because the search will wrap around to the top of the file and continue searching.

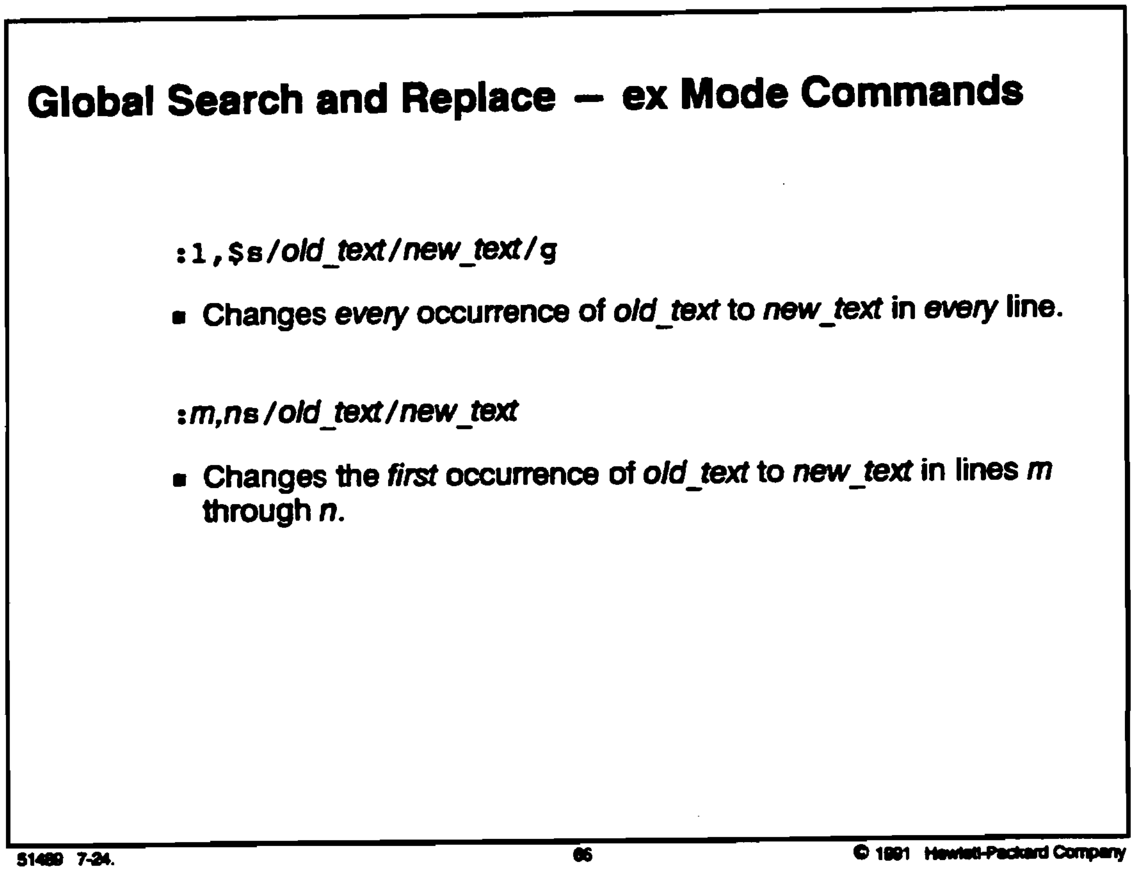
Slide Notes

Uncover each point, in turn explaining what each one does.

Transition

Now let's see how we can globally search and replace text in a file.

7-24. SLIDE: Global Search and Replace—ex Mode Commands



Global Search and Replace – ex Mode Commands

`:1,$s/`*old_text*`/`*new_text*`/g`

- Changes *every* occurrence of *old_text* to *new_text* in *every* line.

`:`*m,n*`s/`*old_text*`/`*new_text*

- Changes the *first* occurrence of *old_text* to *new_text* in lines *m* through *n*.

51489 7-24. 66 © 1991 Hewlett-Packard Company

Student Notes

Global search and replace is performed with an **ex** mode command. **ex** mode is entered when you type :. Whatever you type up to (Return) appears at the bottom of your screen and is interpreted as an **ex** command.

The **ex** mode command to globally search for text and replace it with different text is:

`:1,$s/`*old_text*`/`*new_text*`/g`	Globally substitute *new_text* for *old_text* throughout the file.

The scope of the search can be limited by using line numbers, such as "`5,10`", instead of "`1,$`".

If you omit the g at the end of the command, only the *first* occurrence of *old_text* in each line will be changed.

7-24. SLIDE: Global Search and Replace—ex Mode Commands

Instructor Notes

Purpose

To replace a particular string with a new string in a file.

Key Points

The final g in the expression is used so that *every* occurrence of a particular string in a line will be replaced. If the g is omitted, the command will only apply to the *first* occurrence of a particular string in a line; all other occurrences will remain unchanged.

Where Problems Arise

Some students think that by making the scope of the search every line in the file (`1,$`), they will change *every* occurrence of the string. Stress that a final g must be added to the expression to globally change *every* occurrence in a line.

Evaluation Questions

Will the command `:1,$s/new/old` change every occurrence of `old` to `new`?

Answer: No; only the *first* occurrence of `new` in each line will be changed.

Transition

Here is our last set of exercises in the `vi` module.

7-25. LAB: Searching and Replacing Exercises

1. Using `vi`, edit the file called `new.jersey`. Find the first instance of the word you.

Which way did you move in the file? (Hint: How can we tell what line we are on?)

2. Type `n` once. Where did the cursor go, and in which direction did it travel to get there?

3. Go back to line 1 and search for the first occurrence of the word `the`. What happened?

4. Type `:q` to exit `vi`. Why did it let you out?

5. Using `vi`, edit the file `new.jersey` again. Change all occurrences of `like` to `hate`.

7-25. LAB: Searching and Replacing Exercises

Purpose

You ought to know by now!

Transition

Let's review what we've covered in this module.

Solutions

1. Using `vi`, edit the file called `new.jersey`. Find the first instance of the word you.

Which way did you move in the file? (Hint: How can we tell what line we are on?)

Answer:

/you finds the first occurrence of you by searching downward (toward the end of the file). You can find out the line you are on by typing Ctrl-g.

2. Type n once. Where did the cursor go, and in which direction did it travel to get there?

Answer:

It went to the next instance of you by searching toward the end of the file.

3. Go back to line 1 and search for the first occurrence of the word `the`. What happened?

Answer:

You find `them` instead because `vi` is just looking for the three characters `the` wherever they appear.

4. Type `:q` to exit `vi`. Why did it let you out?

Answer:

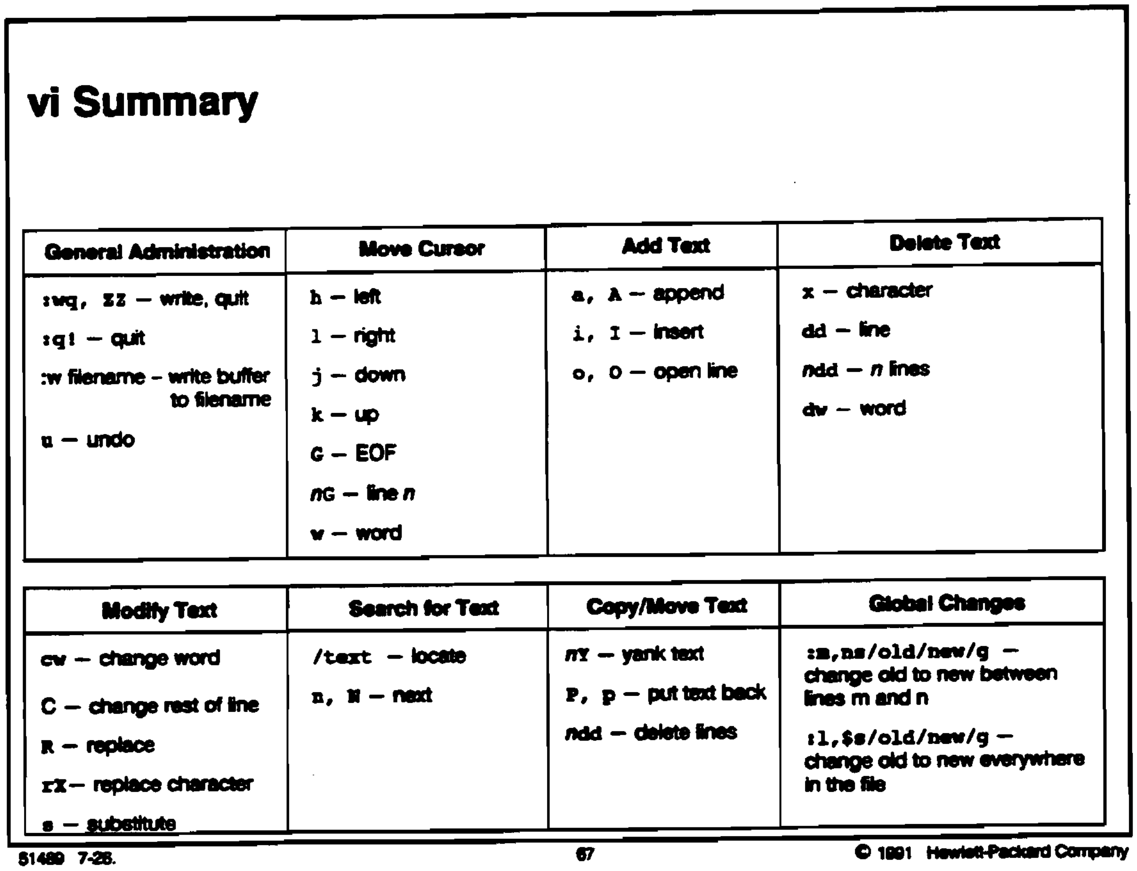
If you do not make changes, `vi` will let you exit a file without saving it first.

5. Using `vi`, edit the file `new.jersey` again. Change all occurrences of `like` to `hate`.

Answer:

Type `:1,$s/like/hate/g`

7-26. SLIDE: vi Summary



vi Summary

General Administration	Move Cursor	Add Text	Delete Text
`:wq`, `ZZ` — write, quit	`h` — left	`a`, `A` — append	`x` — character
`:q!` — quit	`l` — right	`i`, `I` — insert	`dd` — line
:w filename - write buffer to filename	`j` — down	`o`, `O` — open line	*n*`dd` — *n* lines
`u` — undo	`k` — up		`dw` — word
	`G` — EOF		
	n`G` — line *n*		
	`w` — word		

Modify Text	Search for Text	Copy/Move Text	Global Changes
`cw` — change word	`/text` — locate	*n*`Y` — yank text	`:m,ns/old/new/g` — change old to new between lines m and n
C — change rest of line	`n`, `N` — next	`P`, `p` — put text back	`:1,$s/old/new/g` — change old to new everywhere in the file
`R` — replace		*n*`dd` — delete lines	
`rX`— replace character			
`s` — substitute			

51489 7-26. 67 © 1991 Hewlett-Packard Company

Student Notes

To find more about `vi`, you can look in *The Ultimate Guide to the vi and ex Text Editors.* This is a task oriented book that explains how to do specific tasks. It also contains tutorials to speed the learning process. Another source of information is the *HP-UX Reference Manual*, where you can look up the `vi` command.

When you are using a Window, the `vi` commands and procedures are the same as when you are not using a Window. The only difference is that `vi` is contained in the window where you entered the `vi` command.

7-26. SLIDE: vi Summary

Purpose

To review the `vi` topics and commands presented in this module.

Key Points

`vi` is a screen-oriented text editor. It is also an HP-UX command that has its own subset of sophisticated commands.

Presentation Suggestions

You may cover the slide and just uncover the title of each box and let the students tell you what commands should be in each box.

Transition

Now that we know how to use the `vi` editor, let's learn more about the Korn Shell.

Module 8 — The Korn Shell

Objectives

Upon completion of this module, you will be able to do the following:

- Describe what the K-shell is.
- Explain the features of the K-shell.
- Use the K-shell command line recall and editing features.
- Use file name completion.
- Use the K-shell alias feature.
- Set up your environment to customize the K-shell.

Module 8 — The Korn Shell

Overview of Module 8

This module provides an introduction to the Korn shell. It describes the basic functionality of the K-shell user interface. It does not go into details concerning how the K-shell program environment differs from `sh`.

It should take about 1 hour to cover.

Motivation

The Korn Shell offers a lot of features that make it easier for the user to interface with the system. It offers a command history, command line editing, filename completion, and aliasing while being perfectly compatible with the Bourne Shell. In fact, the Korn Shell is a superset of the Bourne Shell.

Exercises

The exercises are located at the back of the module.

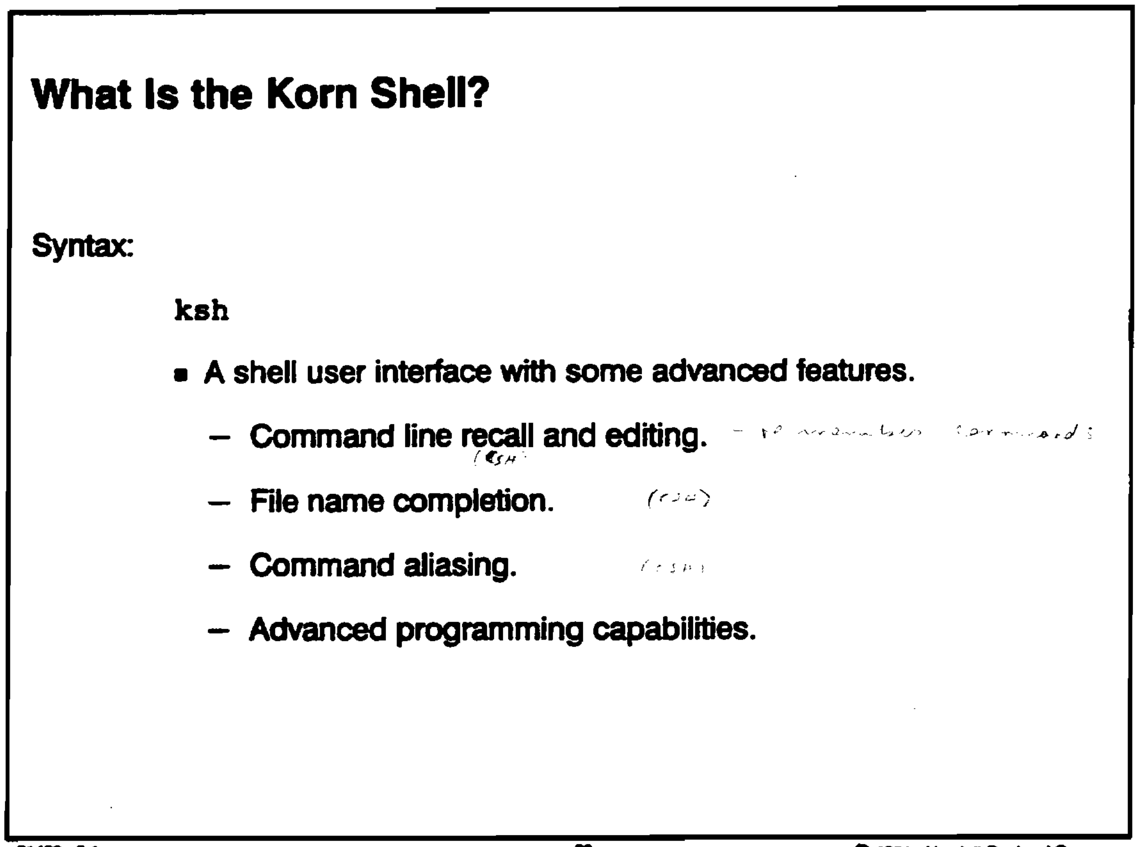
Exercise Files

`$HOME/.profile`, `$HOME/.kshrc`, `cars+dogs`

Module Transition

We have seen commands that produce output and others that can expect input. Let's take a look at some of the different things we can do with the input and output of commands.

8-1. SLIDE: What Is the Korn Shell?



What Is the Korn Shell?

Syntax:

```
ksh
```

- A shell user interface with some advanced features.
  - Command line recall and editing. – [illegible] (csh)
  - File name completion. ([illegible])
  - Command aliasing. ([illegible])
  - Advanced programming capabilities.

51489 8-1. 68 © 1991 Hewlett-Packard Company

Student Notes

One of the shells provided with HP-UX is the **Korn shell** (K-shell). This shell has many features that another shell, the Bourne shell, does not have. Even if you do not use all of the advanced features, you will probably find the Korn shell a very convenient user interface. Here are just a few of the nice features of the K-shell:

- Command line recall and editing.
- File name completion.
- Command aliasing.
- Advanced programming features.

We will discuss some of these features of the K-shell in this module.

The Korn shell was written by David Korn of AT&T Bell Labs.

8-1. SLIDE: What Is the Korn Shell?

Instructor Notes

Purpose

To introduce the Korn shell as a user interface.

Key Points

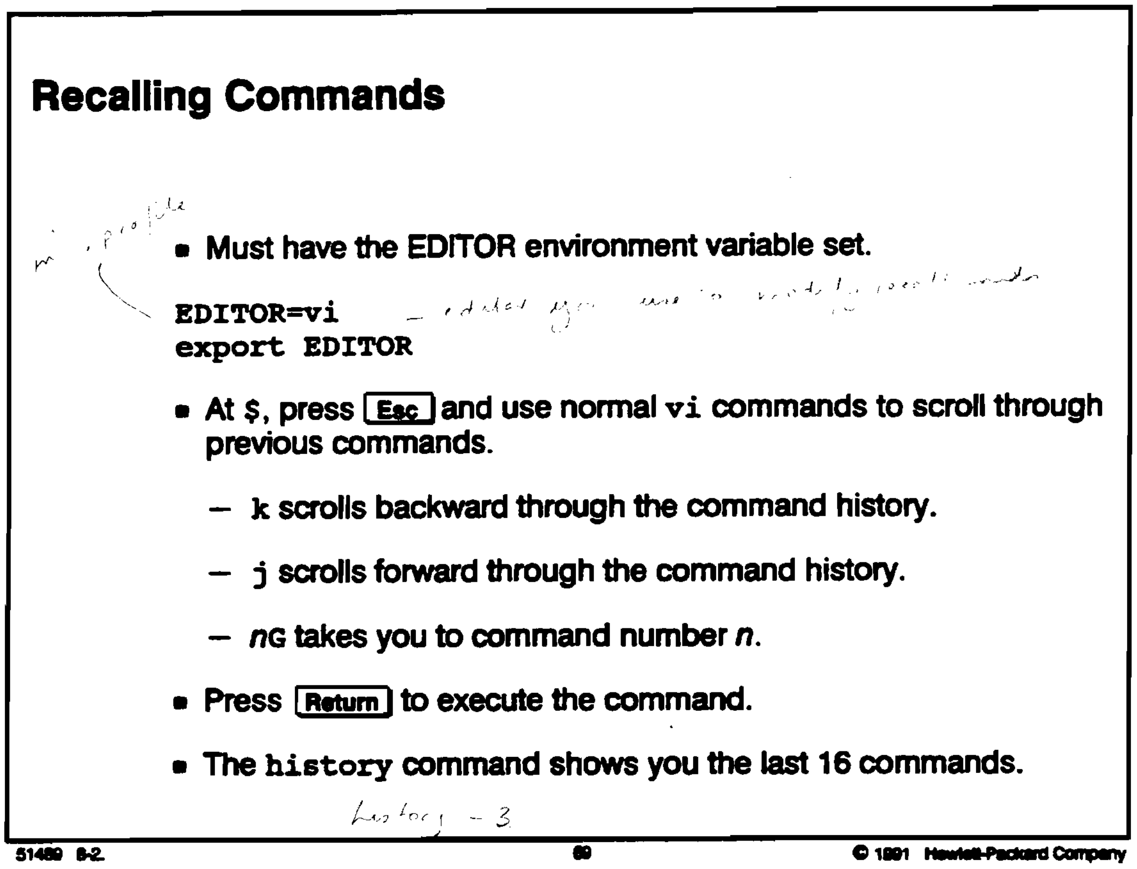
We will discuss command line recall and editing, file name completion, and aliasing in this module.

Transition

Let's take a closer look at these features of the K-shell.

[illegible]

8-2. SLIDE: Recalling Commands



Recalling Commands

in .profile

- Must have the EDITOR environment variable set.

```
EDITOR=vi
export EDITOR
```

— editor you use to [illegible]

- At $, press Esc and use normal `vi` commands to scroll through previous commands.
  - `k` scrolls backward through the command history.
  - `j` scrolls forward through the command history.
  - *n*G takes you to command number *n*.
- Press Return to execute the command.
- The `history` command shows you the last 16 commands.

history -3

51489 8-2 69 © 1991 Hewlett-Packard Company

Student Notes

The K-shell offers a history feature that allows you to recall your previous commands so that you can re-execute them without retyping the line. This history mechanism also allows you to edit previous command lines using the `vi` editor. These features can save you a great deal of typing. If you are not a great typist, they will also save you a lot of time and aggravation.

In order to use the K-shell history mechanism, you need the *EDITOR* variable set in your environment. If you execute the `env` command, you should see this in the listing:

```
$ env
.
.
EDITOR=vi
.
.
```

If this parameter is not set, execute these commands to set it:

```
$ EDITOR=vi
$ export EDITOR
```

This tells the K-shell that you want to use the `vi` editor to recall and edit your previous commands. Put these commands in `.profile` if you want to make sure EDITOR is set every time you log in.

To recall a previous command, simply press Esc. You will not see anything happen on your screen yet. Pressing Esc puts you in K-shell's `vi` mode. At this point you have many of the normal `vi` commands available to you. For example, pressing k moves you back one command in your command stack. If you continue to press k, you will see your previous commands appear on your command line one at a time. Similarly, if you press the j key, you will scroll through your commands in the opposite direction. When you see the command you want to execute again on your command line, just press Return.

Use the `history` command to see your last 16 commands. This will list the number of the command with the command line. If you want to execute a particular command, type Esc *n*`G`, where *n* is the command line number from the `history` listing. Recall that the `G` command in `vi` moves you to a specific line.

8-2. SLIDE: Recalling Commands

Purpose

To introduce the K-shell command recall facility.

Key Points

The *EDITOR* parameter must be set in order to use the recall and editing features of **ksh**. This is normally done in the user's `.profile`.

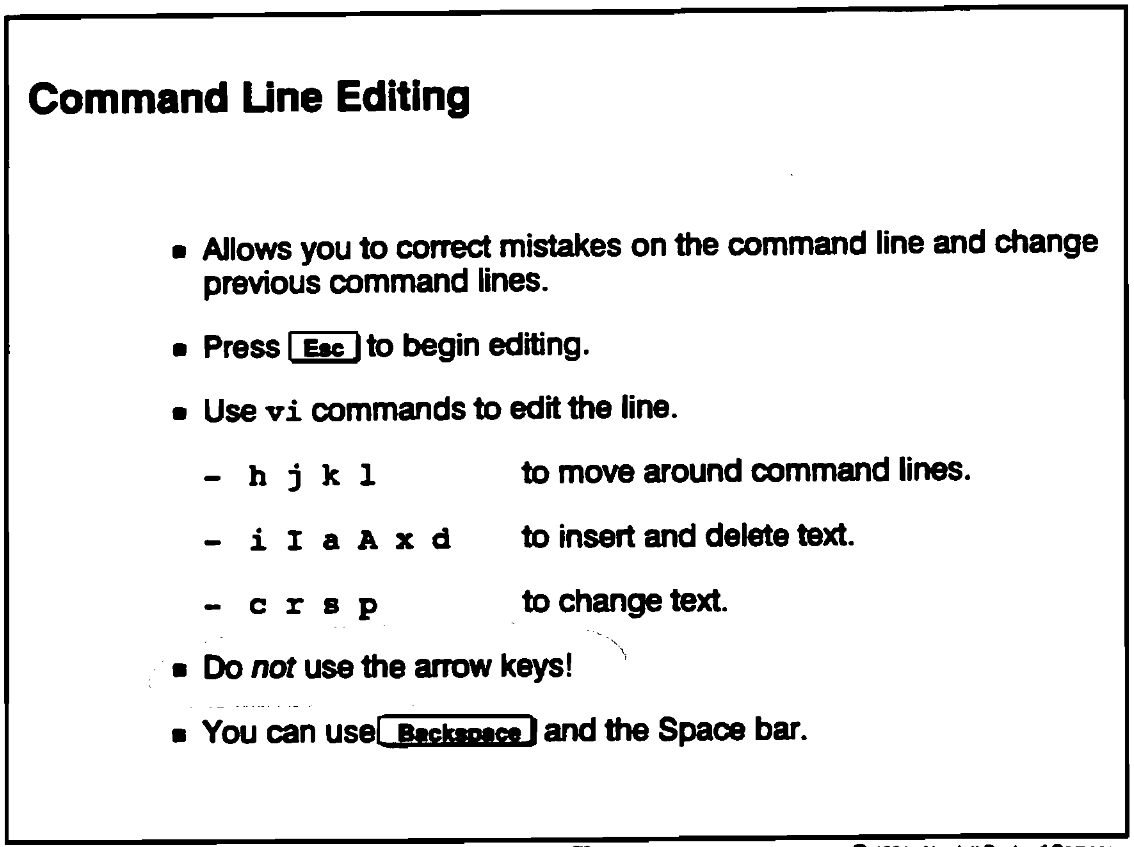
Evaluation Question

What variable must be set in order to use the history mechanism? Answer: *EDITOR*.

Transition

What if we need to change a previous command before we can re-execute it?

8-3. SLIDE: Command Line Editing



Command Line Editing

- Allows you to correct mistakes on the command line and change previous command lines.
- Press Esc to begin editing.
- Use `vi` commands to edit the line.
  - `h j k l` to move around command lines.
  - `i I a A x d` to insert and delete text.
  - `c r s p` to change text.
- Do *not* use the arrow keys!
- You can use Backspace and the Space bar.

51489 8-3. 70 © 1991 Hewlett-Packard Company

Student Notes

How many times have you been typing a long command line when you found out that you made a mistake at the very beginning of the line? It happens all the time, and all you can do is backspace and retype everything after the mistake.

The K-shell lets you correct your mistakes and change parts of a command line before you execute it. Once again, this is done with the `vi` editing commands.

To change a command line, you must press Esc to enter the `vi` editing mode. This works on command lines that you are typing *and* on the lines that you recalled using Esc and k.

Once you are in editing mode, the `vi` commands work. For example, `x` deletes a character, `h` and `l` move you left and right across the line, `cw` changes a word, `dw` deletes a word, and so on.

Here is an example:

```
$ cp /usr/lib/X11/app_defaults
Usage: cp f1 f2
       cp [-r] f1 ... fn d1
```

That was supposed to be cd, not cp. K-shell lets you fix the line without retyping it. Just press Esc and then k and the command line will come back. Type l to move to the p in cp and use the r command to replace the p with a d. Your command line will now look like this:

```
$ cd /usr/lib/X11/app_defaults
```

Now just press Return and the cd command will execute.

If you had problems editing the line and want to try again, just press Break to cancel editing, and you will get your regular shell prompt back so you can try again.

Do not use the arrow keys when you are editing command lines in the K-shell. In addition to the h and l keys, you may use Backspace and the Space bar.

8-3. SLIDE: Command Line Editing

Instructor Notes

Purpose

Command line editing is the most powerful feature of the K-shell.

Key Points

Whether you are typing a command line or you have just recalled a command line, you can edit it using the `vi` editing commands.

Pressing `Break` cancels the editing and puts you back to normal K-shell mode.

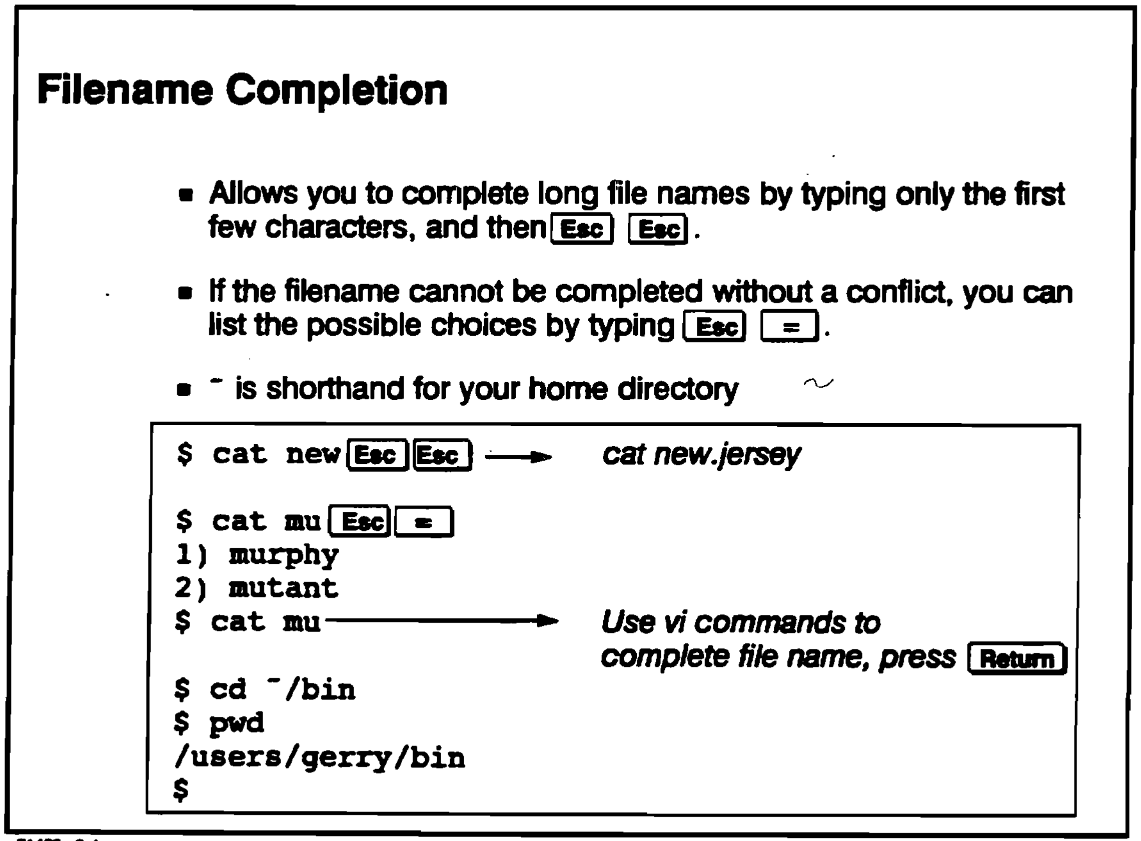
Evaluation Question

Will `vi` commands such as `G`, `D`, `p`, and `s` work? Sure! The results may be a little strange, however, because you can only see one line at a time.

Transition

File name completion helps when you are typing long path names and file names.

8-4. SLIDE: Filename Completion



Filename Completion

- Allows you to complete long file names by typing only the first few characters, and then [Esc] [Esc].
- If the filename cannot be completed without a conflict, you can list the possible choices by typing [Esc] [=].
- ~ is shorthand for your home directory

```
$ cat new[Esc][Esc]  ——►      cat new.jersey

$ cat mu[Esc][=]
1) murphy
2) mutant
$ cat mu ————————————►      Use vi commands to
                            complete file name, press [Return]
$ cd ~/bin
$ pwd
/users/gerry/bin
$
```

51489 8-4. 71 © 1991 Hewlett-Packard Company

Student Notes

It can be annoying when you try to access a file that has a long name and you keep misspelling it. The K-shell has a facility called **file name completion** that will allow you to type only the beginning of a file or directory name and it will complete it for you.

For instance, if you were trying to access the file `supercalifragilistic`, you could just type `sup`[Esc] [Esc]. K-shell will complete a file or directory name when you press [Esc] twice.

There is a trick. What you type must be unique before the K-shell will complete it. For example, if you also have a file in your directory called `superduper`, pressing [Esc] twice will give you `super` and the K-shell will beep to let you know that there is more than one file name that starts with `super`. At this point, you could list the possible choices by typing [Esc] [=]. Then, using the command line editor, you can now complete the file name yourself or add sufficient characters so the shell can complete the file name when you type [Esc] [Esc] again.

You only need to type enough of the name so the K-shell can make a unique match.

Another useful feature of file names is using the ~ character to indicate your home directory. For example:

```
$ ls -F ~/mail
```

will list the contents of the `mail` directory which is under your home directory. This is useful because it works from any directory in the file system.

8-4. SLIDE: Filename Completion

Instructor Notes

Purpose

The file name completion feature is yet another reason the K-shell is so nice to use.

Key Points

Pressing (Esc) twice will make the K-shell complete the file name you are currently typing. If the file name is not unique, the K-shell will beep and display the name up through where the names differ.

"~" means home directory.

Evaluation Questions

What would happen if you typed `more ~/n` (Esc) (Esc)? The K-shell would just beep at you because there is more than one file name starting with `n`. Using `ne` instead would give you `new.jersey` when you pressed (Esc) twice.

Transition

The last feature of the K-shell that we will look at is the alias function.

8-5. SLIDE: Command Aliasing

Command Aliasing

Syntax:

```
alias [name[='command_string']]
```

- Allows you to give a command string a new name.
- Without arguments, it reports all aliases.

```
$ alias laser='lp -dlaser'
$ laser fileX
request id is laser-996 (1 file)
$ alias lsf='ls -Fa'
$ lsf
./      .kshrc     bin/     new.jersey
../     .profile   mail/    testfile
$ alias rm='rm -i'
$
```

51489 8-5.   72

Student Notes

An alias is a new name for a command. The K-shell allows you to create aliases to make your user interface a little more friendly. It also lets you make shorthand versions of commands with many options. For example, many people use the `ps -ef` command quite often. Wouldn't it be much easier if you could type `psf` instead?

You can create aliases in the K-shell using the **`alias`** command. The general form is:

```
$ alias name='command'
```

where *name* is the new name you are using for the alias, and *command* is the command that *name* is aliased to.

Let's say you have an HP LaserJet connected to your system and in order to print to it you would have to type the command `$ lp -dlaser` *filename*. It would be much easier to use an alias like the following:

```
$ alias laser='lp -dlaser'
$ laser filename
```

The "laser" alias would be in effect until you logged out.

You can do this to any command, allowing you to tailor your environment to have the commands you like to type. Many people who work with MS-DOS computers, for instance, like to alias the `ls -l` command to `dir`, `cp` to `copy`, `rm` to `del`, and so on.

To get a list of aliases set up in your shell, use the `alias` command with no arguments. You will see that many commands in the K-shell are really aliases.

To list the value of a particular alias, use `alias` *name*.

8-5. SLIDE: Command Aliasing

Instructor Notes

Purpose

The alias feature is a great way to customize the K-shell to your particular taste.

Key Points

Aliases are only active until you log off. On the next page we will see how to set up aliases at login. The K-shell will only interpret aliases when they are used as commands.

Evaluation Question

How could you alias the `rm` command so it is interactive by default? Answer:

```
$ alias rm='rm -i'
```

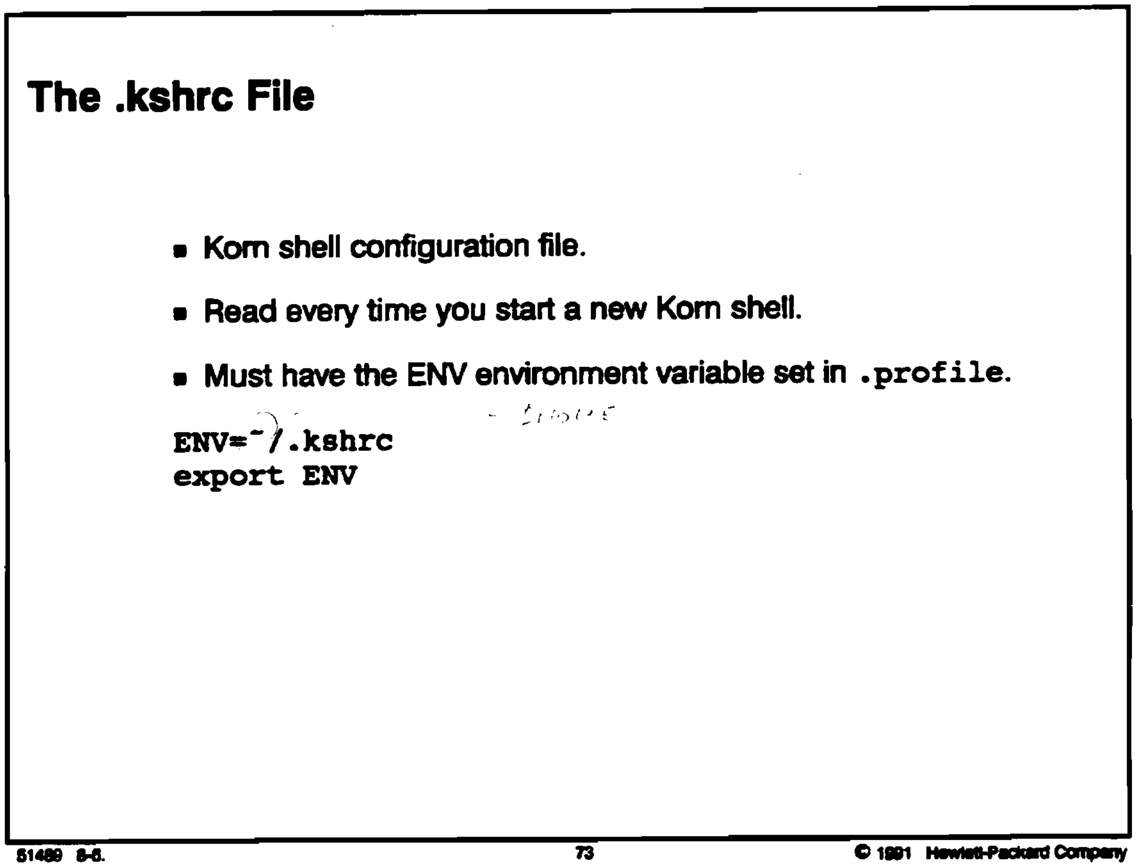
Presentation Suggestions

Students usually love this part. Ask them for some ideas of aliases and how to create them.

Transition

Aliases are only active until we log off. How can we make an alias active when we log in?

8-6. SLIDE: The .kshrc File



The .kshrc File

- Korn shell configuration file.
- Read every time you start a new Korn shell.
- Must have the ENV environment variable set in `.profile`.

~ $HOME

```
ENV=~/.kshrc
export ENV
```

51489 8-6. 73 © 1991 Hewlett-Packard Company

Student Notes

The K-shell has an optional configuration file called `.kshrc`. It is used much like `.profile` to configure your user environment. Unlike, `.profile`, however, `.kshrc` is read every time you start a new shell, not just when you log in. This allows you to set up your aliases or even your prompt every time you start a shell. In an environment like X11 Windows, you may have several shells running at once. You can use the `.kshrc` file so that every one of those shells looks the same.

To use your `.kshrc` file, you must put a new environment variable in your `.profile`. This is the `ENV` variable. Add these lines to your `.profile`:

```
ENV=~/.kshrc
export ENV
```

This tells the K-shell that you want to use the `.kshrc` file in your home directory (`~/.kshrc`).

Now just add all of your alias commands to `.kshrc`. This is a sample of a `.kshrc` file:

```
#!/bin/ksh

# The ^[ is the escape character.  You can insert an escape using vi by
# typing Ctrl-v before the escape.

iv="^[&dB"     # Hewlett-Packard escape sequence for inverse video
nv="^[&d@"     # Hewlett-Packard escape sequence for normal video
flash="^[&dA"  # Hewlett-Packard escape sequence for flashing video

# Change the prompt so it displays the command number in inverse video
PS1="${iv}[!]${nv} $ "
# Comment out the above line and uncomment the next command line
# to get a blinking sideways smiley face in your prompt.
#PS1="${flash}8-)${nv}${iv}[!]${nv} $ "

# Set some handy aliases
alias rm='rm -i'
alias slp='lp -dsyslp'
alias lps='lpstat -t'
alias ls -F='ls -Fa'
alias copy='cp'
alias bye='exit'
```

Using this `.kshrc` file, your prompt would be changed to inverse video and display the command line number for you, and you would have the six aliases set every time you entered the K-shell.

8-6. SLIDE: The .kshrc File

Instructor Notes

Purpose

It would be terribly inconvenient if you had to retype all of your aliases every time you logged in or started a new shell. The `.kshrc` file takes care of this for us.

Key Points

The *ENV* variable must be set in `.profile` or `.kshrc` will never be read.

`.kshrc` is a shell script that is executed every time a K-shell is started.

Evaluation Question

What do the alias commands in this sample `.kshrc` do? Answer: The first four aliases are shorthand versions of commands with options. The last two aliases are new names for commands.

Presentation Suggestion

Go over the aliases and prompt change with the students. They actually have a similar `.kshrc` in their home directories that they will set up in lab.

Transition

It's fun to talk about the neat features of the K-shell, but trying them really shows you the benefits.

8-7. LAB: Exercises

1. Make sure you are in your home directory. What happens when you type `more n` Esc Esc? Using this command line, how can you make it display `new.jersey`.

2. Create an alias called "psf" that executes the `ps -ef` command.

3. Edit your `.profile` to set the *ENV* variable to `~/.kshrc`.

Advanced:

Do this from the directory you are in using the ~ in the path name.

4. Check the commands in the `.kshrc` file in your home directory. Add your psf alias to the list.

5. Log out and then log back in to test your aliases. Why did you have to log out?

6. Type this incorrect command without hitting Return:

```
cd /user/spol/ko/interface
```

Using command line editing, correct the line to read:

```
cd /usr/spool/lp/interface
```

(Do *not* retype the command)

7. Execute the command `ls -F ~/cars+dogs`.

Recall this command line and change the `ls -F` to `ls -l` using whatever `vi` editing commands are necessary. Re-execute the command.

Advanced:

Try the command `ls -F ~/cars+dogs` without typing `cars+dogs` but using file name completion.

7. Execute the command `ls -F ~/cars+dogs`.

Recall this command line and change the `ls -F` to `ls -l` using whatever `vi` editing commands are necessary. Re-execute the command.

Advanced:

Try the command `ls -F ~/cars+dogs` without typing `cars+dogs` but using file name completion.

8-7. LAB: Exercises

1. Make sure you are in your home directory. What happens when you type `more n` (Esc) (Esc)? Using this command line, how can you make it display `new.jersey`.

2. Create an alias called "psf" that executes the `ps -ef` command.

3. Edit your `.profile` to set the *ENV* variable to `~/.kshrc`.

Advanced:

Do this from the directory you are in using the `~` in the path name.

4. Check the commands in the `.kshrc` file in your home directory. Add your psf alias to the list.

5. Log out and then log back in to test your aliases. Why did you have to log out?

6. Type this incorrect command without hitting (Return):

```
cd /user/spol/ko/interface
```

Using command line editing, correct the line to read:

```
cd /usr/spool/lp/interface
```

(Do *not* retype the command)

8-7. LAB: Exercises

Instructor Notes

Purpose

To practice some of the features of the K-shell.

Where Problems Arise

If any students are working in VUE, they must edit `.vueprofile`, instead of `.profile`, in exercise 3.

Transition

Let's learn more about what we can do with files.

Solutions

1. Make sure you are in your home directory. What happens when you type `more n` Esc Esc? Using this command line, how can you make it display `new.jersey`.

Answer:

Typing the command line given puts `more n` on the command line, and the K-shell beeps because there is more than one file starting with `n`. If you type an `e` and then Esc Esc again, the file name `new.jersey` will be completed for you.

2. Create an alias called "psf" that executes the `ps -ef` command.

Answer:

```
$ alias psf='ps -ef'
```

3. Edit your `.profile` to set the *ENV* variable to `~/.kshrc`.

Advanced:

Do this from the directory you are in using the ~ in the path name.

Answer:

You would add these two lines to your `.profile`:

```
ENV=~/.kshrc
export ENV
```

4. Check the commands in the `.kshrc` file in your home directory. Add your psf alias to the list.

5. Log out and then log back in to test your aliases. Why did you have to log out?

Answer:

You had to reread the `.profile` and `.kshrc` files. The easiest way is to log out and then log back in.

6. Type this incorrect command without hitting Return:

```
cd /user/spol/ko/interface
```

Using command line editing, correct the line to read:

```
cd /usr/spool/lp/interface
```

(Do *not* retype the command)

Answer:

```
$ cd /user/spol/ko/interface Esc
```

Using Backspace and the space bar to position the cursor, use `vi` commands `x`, `a`, `cw` to make the appropriate changes. Remember to use Esc whenever you need to leave input mode.

7. Execute the command `ls -F ~/cars+dogs`.

Recall this command line and change the `ls -F` to `ls -l` using whatever `vi` editing commands are necessary. Re-execute the command.

Advanced:

Try the command `ls -F ~/cars+dogs` without typing `cars+dogs` but using file name completion.

Answer:

```
$ ls -F ~/cars+dogs
car.models/    collie    dog.breeds/    poodle
$ Esc k
```

Now use the `r` command to change `ls -F` to `ls -l` and press Return.

```
$ ls -F ~/ca Esc Esc
```

Module 9 — Changing Input and Output

Objectives

Upon completion of this module, you will be able to do the following:

- Explain the concept of input and output redirection and pipelines.
- Explain the meaning of stdin and stdout.
- Use output redirection to create files.
- Use pipes to connect commands.

Module 9 — Changing Input and Output

Overview of Module 9

I/O redirection allows commands to read input from a file or save output to a file. The concept of redirection may be a little confusing. Output redirection is usually pretty straightforward. Input redirection, however, is something that takes a little more time to grasp. Although input redirection is not used as often as output redirection, the concept of commands using stdin is important to using pipes.

Motivation

Most commands that the students have seen that produce output display it on the terminal, and some, like the editor, take their input from the terminal. Redirection allows a user to control the input to and the output from a shell command.

One of the most powerful features of the UNIX operating system is the ability to connect several simple commands into one more complex command string using pipes.

As students use these facilities, they will quickly learn their usefulness.

Exercises

There are exercises throughout this short module. They are positioned in the module such that they help to reinforce a new concept.

Exercise Files

`new.jersey`

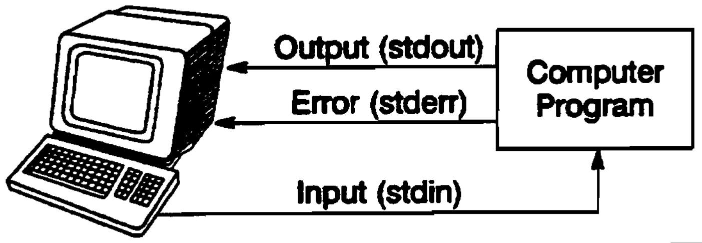
Module Transition

The line printer facility is used quite often on UNIX systems. We will take a look at it next. You can use I/O redirection and pipes with the line printer.

9-1. SLIDE: Input and Output Redirection

Input and Output Redirection

- Output redirection.
  - For commands that produce output, we can redirect the output to go somewhere other than the terminal's display.
- Input redirection.
  - For commands that read the terminal's keyboard for input, we can redirect the input to come from somewhere other than the keyboard.
- Usually, the keyboard is standard input and the display is standard output and standard error.



Student Notes

Some commands operate on your files but produce no output on your display. `cp`, `mv`, and `rm` are a few examples. Many commands that we have seen, however, do produce output on the display. `ls`, `more`, `who`, and `banner` are just a few examples. We say these commands write their messages on **standard output** (stdout). stdout, by default, is your terminal's display. If you want to save the output of a command in a file, you can redirect stdout to go somewhere other than your display. This is called **output redirection.**

An analogy is a clerk's out-basket. Normally, the clerk does some work and places the result in the out-basket (stdout). The out-basket is the clerk's standard output. If the clerk puts the work in a file folder instead of putting it in the out-basket, then we can say the clerk has redirected the output.

We have seen commands that produce error messages. These messages are written to **standard error** (stderr). By default, stderr is also your terminal's display. You can use output redirection to save error messages in a file or to just throw them away.

Many commands can read input from your terminal's keyboard. Your keyboard is also **standard input** (stdin) for these commands. When we talk about input redirection, we mean that we are going to make the command read its input from a file instead of from the keyboard.

Again we can use the clerk analogy. Normally, the clerk does work that is put in the in-basket (stdin). If the clerk goes to a file folder to get work to do, we can say that the clerk has redirected the input.

This method of manipulating input and output is called **I/O redirection.**

9-1. SLIDE: Input and Output Redirection — Instructor Notes

Purpose

To introduce the concepts of stdout and stdin and the concept of redirecting stdout and stdin.

Key Points

The terminal's display is stdout for commands that produce output on the display. We can redirect stdout to a file instead.

The keyboard is stdin for some commands. Using input redirection, we can make the command read its input from a file. We have not yet discussed how commands use stdin.

Evaluation Question

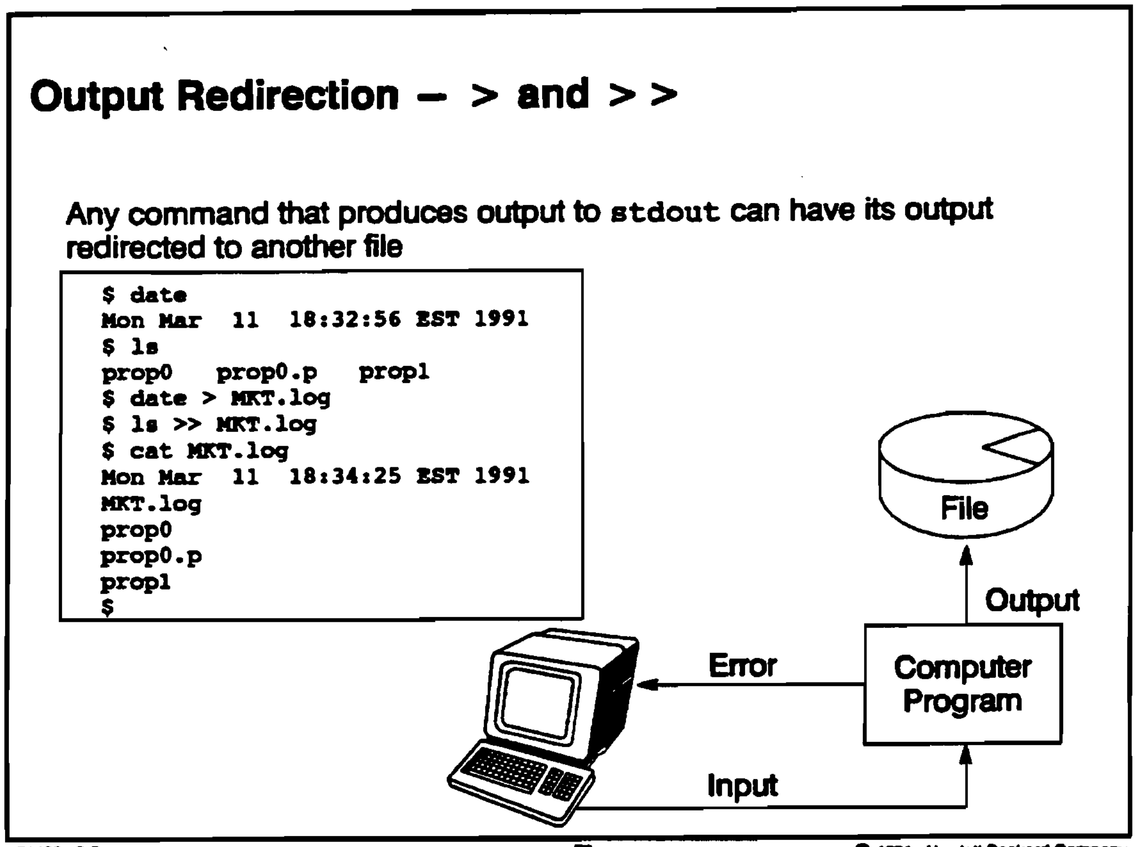
What commands have we seen that write their output to stdout? Answer: `who`, `date`, `banner`, `ls`, `cat` are some commands.

What commands have we seen that read their input from stdin? Answer: `mailx` reads its input from stdin.

Transition

This is an example of how you can use output redirection.

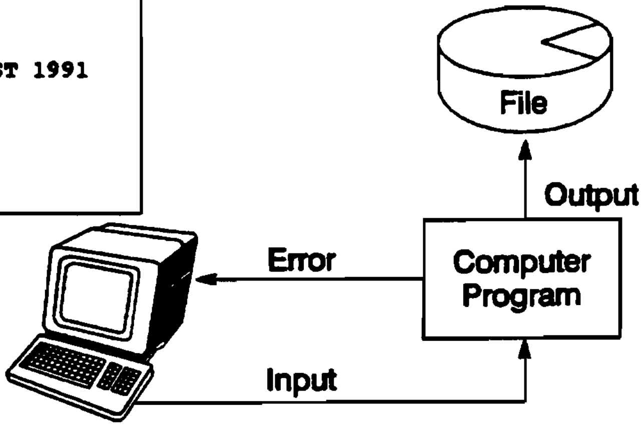
9-2. SLIDE: Output Redirection — > and >>



Output Redirection – > and > >

Any command that produces output to `stdout` can have its output redirected to another file

```
$ date
Mon Mar  11  18:32:56 EST 1991
$ ls
prop0   prop0.p   prop1
$ date > MKT.log
$ ls >> MKT.log
$ cat MKT.log
Mon Mar  11  18:34:25 EST 1991
MKT.log
prop0
prop0.p
prop1
$
```



51489 9-2 75 © 1991 Hewlett-Packard Company

Student Notes

program must normally output to the screen

To redirect stdout from a command to a file, you must include > *filename* on the command line. The symbol ">" indicates that you want to redirect the output of the command to *filename*. Think of the ">" as a funnel.

If *filename* did not exist before the command was invoked, then the file would be created containing the command output. If *filename* did exist, however, it would be overwritten with the output of the command.

```
$ who > logfile
$ cat logfile
doug       console      Jul  7 08:47
jimd       tty0p3       Jul  7 08:49
john       pty/ttyp2    Jul  7 08:59
$
```

Notice that the `who` command did not produce any output on the screen because we redirected it to `logfile`. The `cat` command verifies that `logfile` contains the output of the `who` command.

Suppose you did not want to overwrite a file with the output? Instead, you wanted to append a command's output to a file. To append stdout to a file, you can use >> *filename* on the command line. The symbol ">>" means *append* the output to *filename*. Notice there are no spaces in ">>". In this case, if *filename* existed before you invoked the command, the output of the command would be added to the end of the file. If *filename* did not exist, it would be created and it will contain the output of the command.

Now we can do things such as the following to keep running log files:

```
$ date >> logfile
$ who >> logfile
$ cat logfile
gerry       console      Jul 17 08:47
jimd        tty0p3       Jul 17 08:49
root        pty/ttyp2    Jul 17 08:59
Thu Jul 25 07:37:03 EDT 1991
gerry       tty1p4       Jul 27 06:43
chip        tty3p2       Jul 27 16:07
```

9-2. SLIDE: Output Redirection — > and >> Instructor Notes

Purpose

To show and explain the two methods of output redirection.

Key Points

> *filename* can appear anywhere on the command line; however, it is usually placed at the end of the line, as shown in the examples.

Positive Instance

Output redirection and appending allow us to keep a log.

Negative Instance

It is important to note that if *filename* exists when you execute the command, it will be overwritten.

Teaching Question

What are some things *you* would use output redirection for?

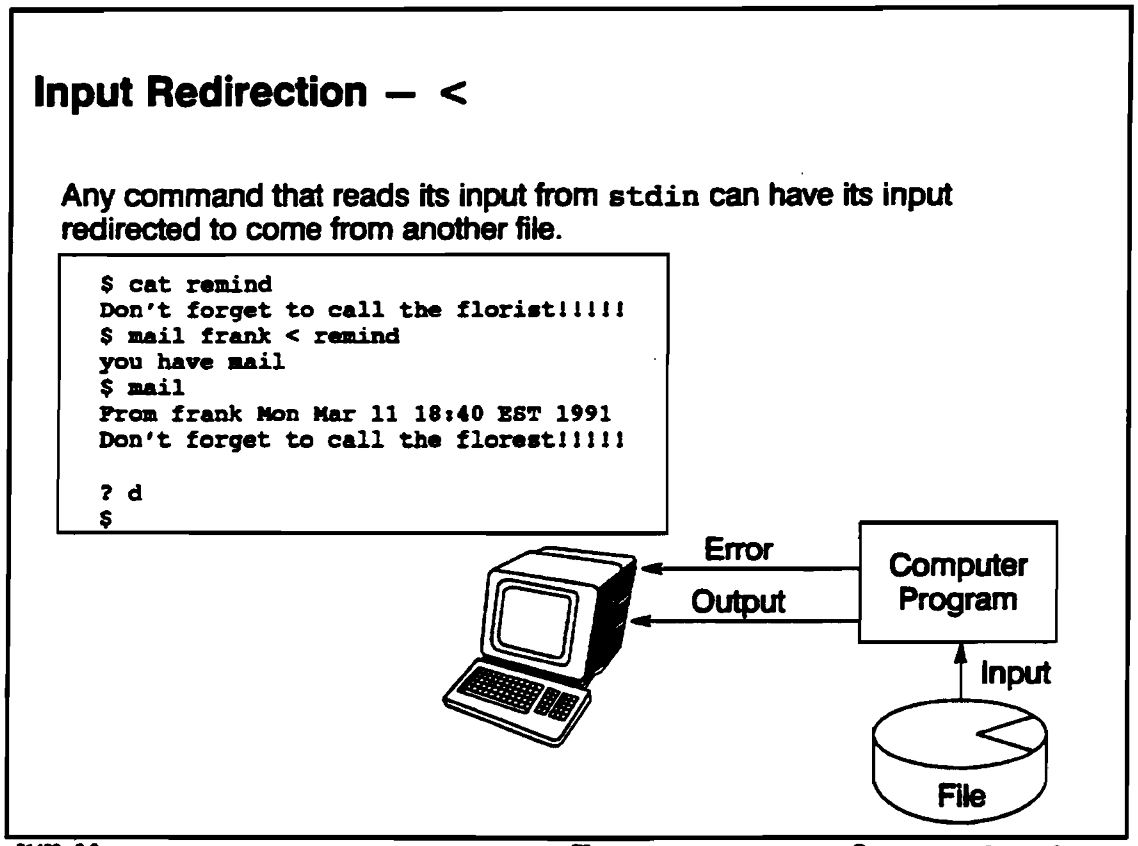
Presentation Suggestion

Use some of the students' examples to show how to use output redirection.

Transition

We can also redirect input to a command instead of letting it read from our terminal.

9-3. SLIDE: Input Redirection — <

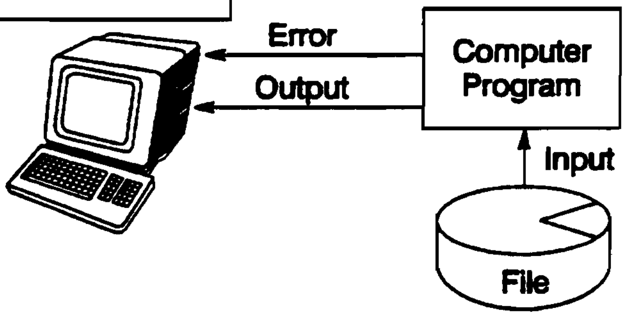


Input Redirection – <

Any command that reads its input from `stdin` can have its input redirected to come from another file.

```
$ cat remind
Don't forget to call the florist!!!!!
$ mail frank < remind
you have mail
$ mail
From frank Mon Mar 11 18:40 EST 1991
Don't forget to call the florest!!!!!

? d
$
```



51480 9-3. 78 © 1991 Hewlett-Packard Company

Student Notes

For commands that take their input from stdin (your keyboard), you can redirect the input from a file instead. To perform input redirection you must include < *filename* on the command line. If *filename* does not exist, you will get an error message on your screen.

The slide shows how you could use input redirection with the `mail` command to send a file instead of typing the message at the terminal.

Many commands that take file names as arguments also read stdin. The `cat` and `more` commands are examples of this type of command. For example, the command `cat new.jersey` is identical to `cat < new. jersey`.

For this reason, input redirection is not used as often as output redirection. We will see other uses for stdin when we talk about pipes. If you start programming in the UNIX operating system or the shell, you will find that stdin and input redirection are used quite often.

Exercise

1. What is the difference between `more new.jersey` and `more < new.jersey`? Try both and notice the slight difference.

9-3. SLIDE: Input Redirection — <

Instructor Notes

Purpose

To introduce the concept of input redirection.

Key Points

Input redirection is not used as often as output redirection because most commands can use a file name as an argument.

Teaching Questions

Why would you want to take input from a file instead of the terminal?

Answer: You may want to mail a message that had already been created with an editor.

`vi` reads its editing commands from stdin. How would input redirection be useful with `vi`?

Answer: You could put `vi` editing commands in a file.

Positive Instance

Students will realize that they can do batch editing using `vi`. They can also create command files if they have programs that normally read the terminal keyboard.

Negative Instance

Many commands do not read stdin. `banner < file` does not work!

Transition

If stderr is a form of output from a command, can we redirect this to a file?

Answer

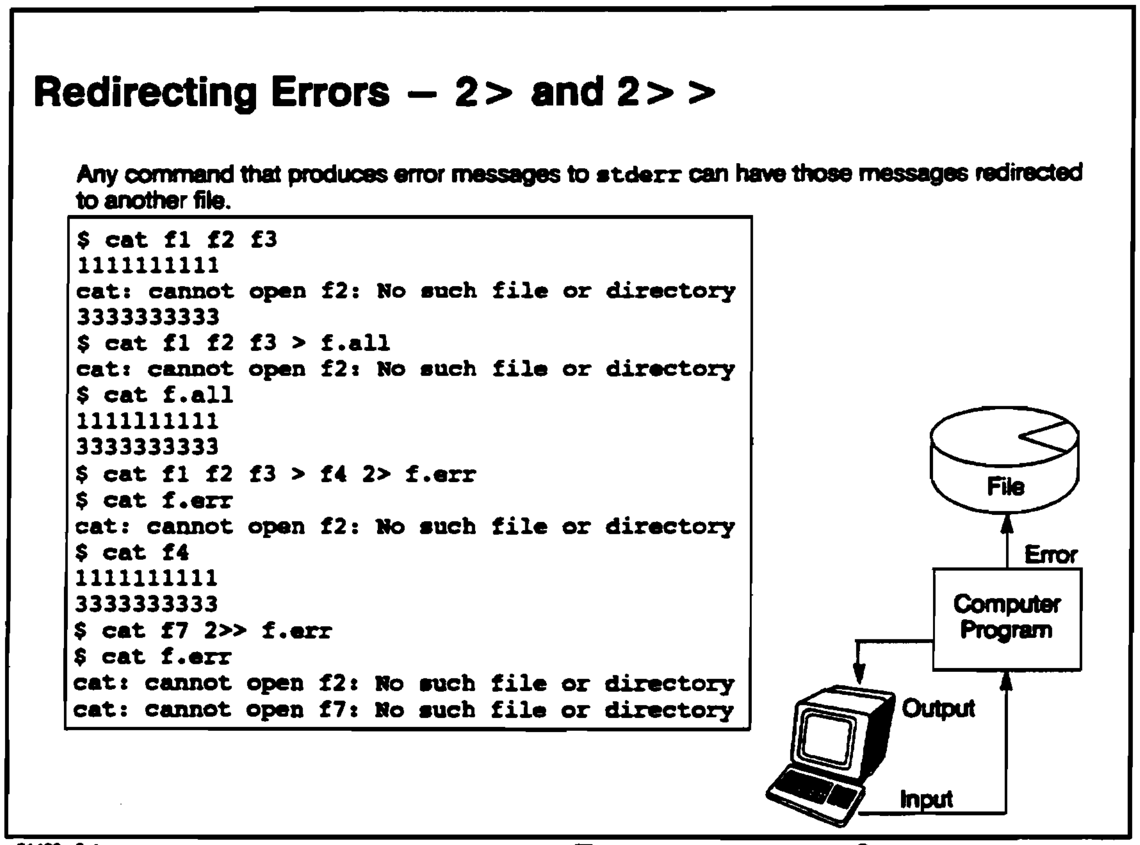
1. What is the difference between `more new.jersey` and `more < new.jersey`? Try both and notice the slight difference.

Answer:

In the command line `more new.jersey`, `more` is reading the file `new.jersey`, while in `more < new.jersey`, `more` is reading stdin, but we have redirected input from `new.jersey`

In the first case, `more` knows the file name and size, so it can print a percentage at the bottom of each screen. When `more` reads stdin, it has no idea how large its input is, so it just displays one screen at a time until the input stops. Because it does not know how large the input is, it cannot display the percentage at the bottom of each page.

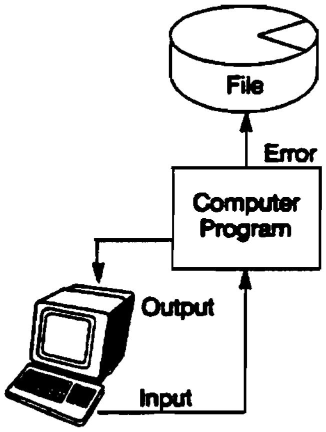
9-4. SLIDE: Redirecting Errors — 2> and 2>>



Redirecting Errors – 2> and 2> >

Any command that produces error messages to `stderr` can have those messages redirected to another file.

```
$ cat f1 f2 f3
1111111111
cat: cannot open f2: No such file or directory
3333333333
$ cat f1 f2 f3 > f.all
cat: cannot open f2: No such file or directory
$ cat f.all
1111111111
3333333333
$ cat f1 f2 f3 > f4 2> f.err
$ cat f.err
cat: cannot open f2: No such file or directory
$ cat f4
1111111111
3333333333
$ cat f7 2>> f.err
$ cat f.err
cat: cannot open f2: No such file or directory
cat: cannot open f7: No such file or directory
```



51489 9-4. 77 © 1991 Hewlett-Packard Company

Student Notes

What happens if you try to cat a file that does not exist? You will get an error message telling you there is a problem. The same is true if you use the cp command with the wrong number of arguments. You can redirect these error messages to a file. This is useful if you want only the output to come to your screen but no error messages. It is also great for saving errors if you are not watching your program run.

To redirect stderr, we use 2>*filename*. There must be no white space between the 2 and the ">".

Notice that on the slide we are trying to cat three files: `f1`, `f2`, and `f3`. However, only `f1` and `f3` exist. `f1` contains a line of 1's, and `f3` contains a line of 3's. When we try to cat `f2`, we get the `cat: cannot open f2` error message written on stderr.

We can append stderr to a file using 2>> (no spaces!).

10-6. SLIDE: LP Spooler System Summary

Instructor Notes

Purpose

The lp spooler system contains a large number of programs and files. We will review the important points of the module before proceeding.

Presentation Suggestions

Have the students explain each of the commands listed on the slide. For each, the students should state the purpose of the command and provide an example of how it would be used. Ask for any remaining questions before proceeding.

Transition

Let's do some exercises now.

10-7. LAB: Exercises

1. List the current status of the printers in the lp spooler system and find the name of the default printer.

2. Send the file named `new.jersey` to the line printer. Make a note of the request id that is displayed on your terminal.

3. Using a pipe, send the output of a long listing of your home directory to the printer. Submit the request so it will print with the title "My files" on the banner page.

4. Verify that your requests are queued to be printed.

5. How can you tell what files other users are printing? Try it.

6. Use the `cancel` command to remove your requests from the line printer system queue. Confirm that they were canceled.

7. Advanced: `pr` is a command that formats output for a printer. It will print a header on each page which includes the date, file name, and page number. Try to `pr` the file `new.jersey`. (You may need to pipe it to `more` to see the output). How could you use `pr` with `lp`?

10-7. LAB: Exercises

Instructor Notes

Purpose

Practice using commands to interact with the lp system.

Key Points

Be sure to enable the system default printer before students begin the exercises.

Transition

We have finished our look at the lp spooler system. Now we will take a look at the user environment.

Solutions

1. List the current status of the printers in the lp spooler system and find the name of the default printer.

Answer:

```
$ lpstat -t
scheduler is running
system default destination: rw
device for rw: /dev/rw
rw accepting requests since Tue Jun 04 10:56:20 1991
printer rw is idle.  enabled since Thu Jun 27 14:32:52 1991
        fence priority : 0
```

2. Send the file named `new.jersey` to the line printer. Make a note of the request id that is displayed on your terminal.

Answer:

```
$ lp new.jersey
request id is rw-58 (1 file)
```

3. Using a pipe, send the output of a long listing of your home directory to the printer. Submit the request so it will print with the title "My files" on the banner page.

Answer:

```
$ ls -l | lp -t"My files"
request id is rw-59 (standard input)
```

4. Verify that your requests are queued to be printed.

Answer:

```
$ lpstat
rw-58              ralph              3967   Thu Jul 04 12:57:25 1991
rw-59              ralph              1331   Tue Jul 02 13:01:19 1991
```

5. How can you tell what files other users are printing? Try it.

Answer:

You can tell by using `lpstat -t`.

6. Use the `cancel` command to remove your requests from the line printer system queue. Confirm that they were canceled.

Answer:

```
$ cancel rw-58  rw-59
request "rw-58" canceled
request "rw-59" canceled
$ lpstat
$
```

7. Advanced: `pr` is a command that formats output for a printer. It will print a header on each page which includes the date, file name, and page number. Try to `pr` the file `new.jersey`. (You may need to pipe it to `more` to see the output). How could you use `pr` with `lp`?

Answer:

```
$ pr new.jersey | more
Dec 11 13:03 1990  new.jersey Page 1


                    I LIKE NEW JERSEY BECAUSE...

$ pr new.jersey | lp
request id is lp-355 (standard input)
```

9-4. SLIDE: Redirecting Errors — 2> and 2>> Instructor Notes

Purpose

We do not always want error messages appearing on the screen in our output. This gives us a way to put it somewhere else.

Key Points

The 2 in 2> means stderr. What we didn't tell the students is that 1 stands for stdout. This means that ">" is really shorthand for 1>. You might also point out 2>&1, which sends error messages to the same place as the standard output.

Teaching Question

When would you use 2>>?

Answer: To keep a log of error messages output by a particular application that is run periodically.

Positive Instance

Use 2>> to keep a log of errors.

Evaluation Question

If we are redirecting stdout, where is stderr displayed?

Where Problems Arise

This slide usually clears up problems with stdout, but the use of stderr may not be clear yet. If there is a problem, it is usually because students cannot grasp why there are two different ways to send messages to the same terminal.

Examples of output and error redirection help clear up most problems.

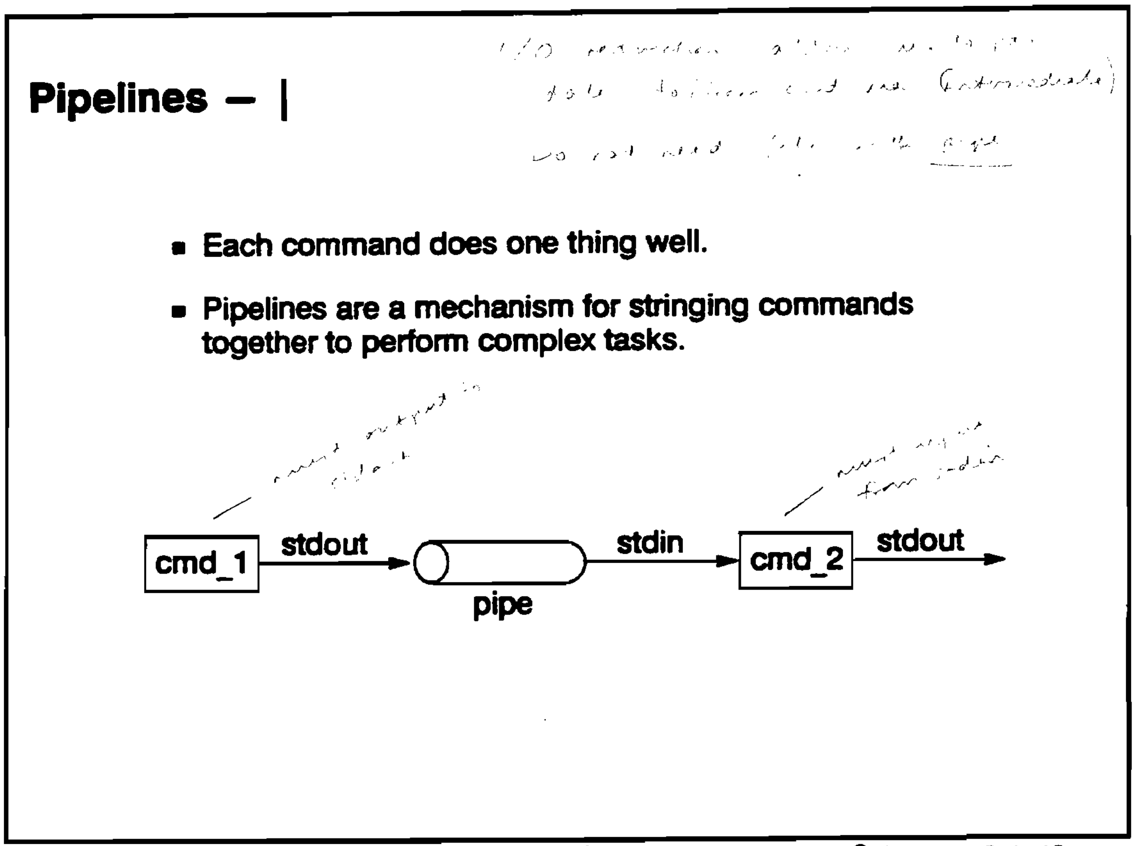
Presentation Suggestions

Go over the slide carefully. You may have to add a few examples of your own.

Transition

Input and output redirection allow you to pass information from a file to a command or from a command to a file. **Pipes** allow us to pass information directly from one command to another.

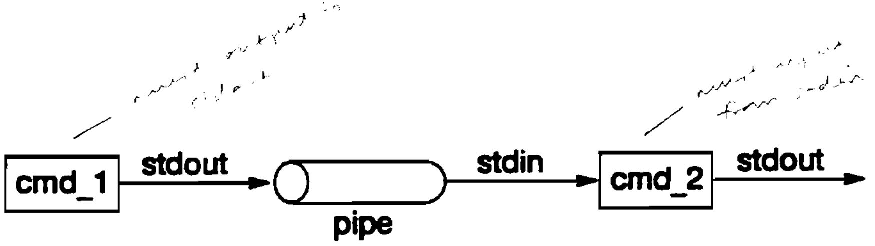
9-5. SLIDE: Pipelines — |



Pipelines — |

[illegible]

- Each command does one thing well.
- Pipelines are a mechanism for stringing commands together to perform complex tasks.



51489 9-5. 78 © 1991 Hewlett-Packard Company

Student Notes

Part of the philosophy of the UNIX operating system was to provide a set of commands in which each one does one thing well. Pipes allow you to string commands together to perform more complex tasks with the simpler commands.

The pipe symbol ("|") is a mechanism that allows you to take the output of one command and run it directly into the input of another command. This is a typical pipeline:

```
$ command1 | command2
```

command1 on the left of the "|" must be producing output on stdout. This means it would normally display something on your screen. *command2* on the right of the "|" must be reading stdin. The | in the middle connects the output of *command1* to the input of *command2*.

[illegible]

9-5. SLIDE: Pipelines — |

Instructor Notes

Purpose

To introduce the concept of a pipe and show a typical use of one.

Key Points

The command on the left of the pipe must produce output on stdout and the command on the right must read input from stdin.

Evaluation Question

Ask the students how "*command1* | *command2*" may be done using I/O redirection.

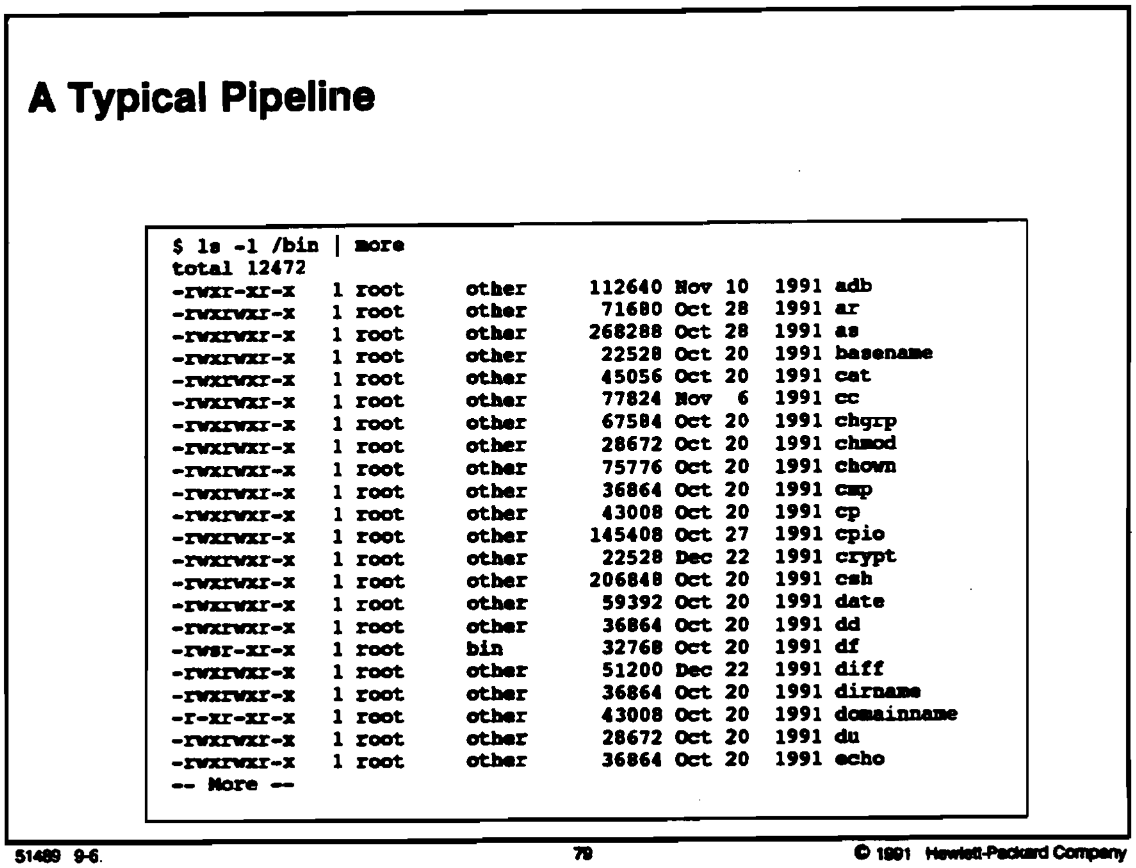
Answer: *command1* > *temp_file*

command2 < *temp_file*

Transition

Here is an example of a typical pipeline.

9-6. SLIDE: A Typical Pipeline



A Typical Pipeline

```
$ ls -l /bin | more
total 12472
-rwxr-xr-x   1 root     other      112640 Nov 10  1991 adb
-rwxrwxr-x   1 root     other       71680 Oct 28  1991 ar
-rwxrwxr-x   1 root     other      268288 Oct 28  1991 as
-rwxrwxr-x   1 root     other       22528 Oct 20  1991 basename
-rwxrwxr-x   1 root     other       45056 Oct 20  1991 cat
-rwxrwxr-x   1 root     other       77824 Nov  6  1991 cc
-rwxrwxr-x   1 root     other       67584 Oct 20  1991 chgrp
-rwxrwxr-x   1 root     other       28672 Oct 20  1991 chmod
-rwxrwxr-x   1 root     other       75776 Oct 20  1991 chown
-rwxrwxr-x   1 root     other       36864 Oct 20  1991 cmp
-rwxrwxr-x   1 root     other       43008 Oct 20  1991 cp
-rwxrwxr-x   1 root     other      145408 Oct 27  1991 cpio
-rwxrwxr-x   1 root     other       22528 Dec 22  1991 crypt
-rwxrwxr-x   1 root     other      206848 Oct 20  1991 csh
-rwxrwxr-x   1 root     other       59392 Oct 20  1991 date
-rwxrwxr-x   1 root     other       36864 Oct 20  1991 dd
-rwsr-xr-x   1 root     bin         32768 Oct 20  1991 df
-rwxrwxr-x   1 root     other       51200 Dec 22  1991 diff
-rwxrwxr-x   1 root     other       36864 Oct 20  1991 dirname
-r-xr-xr-x   1 root     other       43008 Oct 20  1991 domainname
-rwxrwxr-x   1 root     other       28672 Oct 20  1991 du
-rwxrwxr-x   1 root     other       36864 Oct 20  1991 echo
-- More --
```

51489 9-6. 79 © 1991 Hewlett-Packard Company

Student Notes

Pipelines are useful for many reasons. This page shows you just one example of how a pipeline could be used.

The `ls -l` command produces a directory listing and nothing else. What if the directory you were listing was very large and the output of the `ls -l` command scrolled off the screen? You could redirect the output of the `ls -l` command to a file and then use `more` to display the contents of the file.

```
$ ls -l /bin > listing
$ more listing
total 12472
-rwxr-xr-x    1 root     other      112640 Nov 10  1991 adb
-rwxrwxr-x    1 root     other       71680 Oct 28  1991 ar
-rwxrwxr-x    1 root     other      268288 Oct 28  1991 as
-rwxrwxr-x    1 root     other       22528 Oct 20  1991 basename
-rwxrwxr-x    1 root     other       45056 Oct 20  1991 cat
.
.
The directory listing
one screenful at a time
.
```

You could save yourself work and the trouble of creating a temporary file, however, by connecting the output of the `ls -l` command to the input of the `more` command using a pipe (recall that `more` reads stdin):

```
$ ls -l /bin | more
total 12472
-rwxr-xr-x    1 root     other      112640 Nov 10  1991 adb
-rwxrwxr-x    1 root     other       71680 Oct 28  1991 ar
-rwxrwxr-x    1 root     other      268288 Oct 28  1991 as
-rwxrwxr-x    1 root     other       22528 Oct 20  1991 basename
-rwxrwxr-x    1 root     other       45056 Oct 20  1991 cat
.
.
.
The directory listing
one screenful at a time
```

Now the output of the `ls -l` command is read by `more`, which will display it on your terminal one screen at a time.

Pipes are one of the most powerful features of the UNIX operating system.

9-6. SLIDE: A Typical Pipeline

Instructor Notes

Purpose

To give an example of a pipeline.

Key Points

`ls -l` produces output, and `more` will read stdin when it is not given a file name argument. This allows them to work together well in a pipeline.

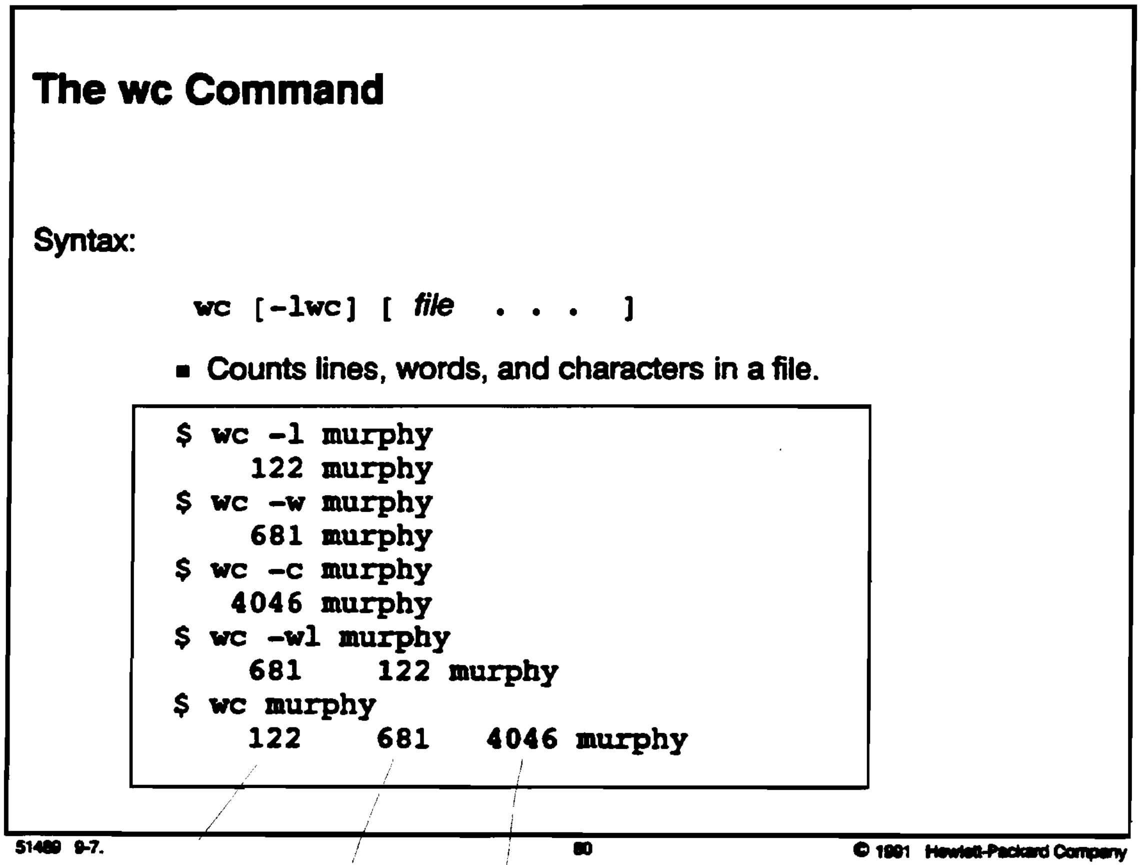
Evaluation Question

Can `more` display a percentage when it is used on the right of a "|"? Answer: No!

Transition

There is another command that is very useful in a pipeline.

9-7. SLIDE: The wc Command



The wc Command

Syntax:

```
wc [-lwc] [ file . . . ]
```

- Counts lines, words, and characters in a file.

```
$ wc -l murphy
   122 murphy
$ wc -w murphy
   681 murphy
$ wc -c murphy
  4046 murphy
$ wc -wl murphy
   681    122 murphy
$ wc murphy
   122    681   4046 murphy
```

51489 9-7. 80 © 1991 Hewlett-Packard Company

[illegible] words chars

Student Notes

The command wc reads its input from stdin and produces output to stdout. Such a command is known as a **filter**.

The wc command is a counter. It counts the number of lines, words, and characters in a file. The command has options -l, -w, and -c. The -l option will display the number of lines; the -w option will display the number of words; and the -c option will display the number of characters.

The order of the options determines the order of the output.

[illegible]

[illegible]

9-7. SLIDE: The wc Command

Instructor Notes

Purpose

To introduce a command that is very useful in a pipeline.

Key Points

Point out that the order of the options determines the order of the output.

Teaching Question

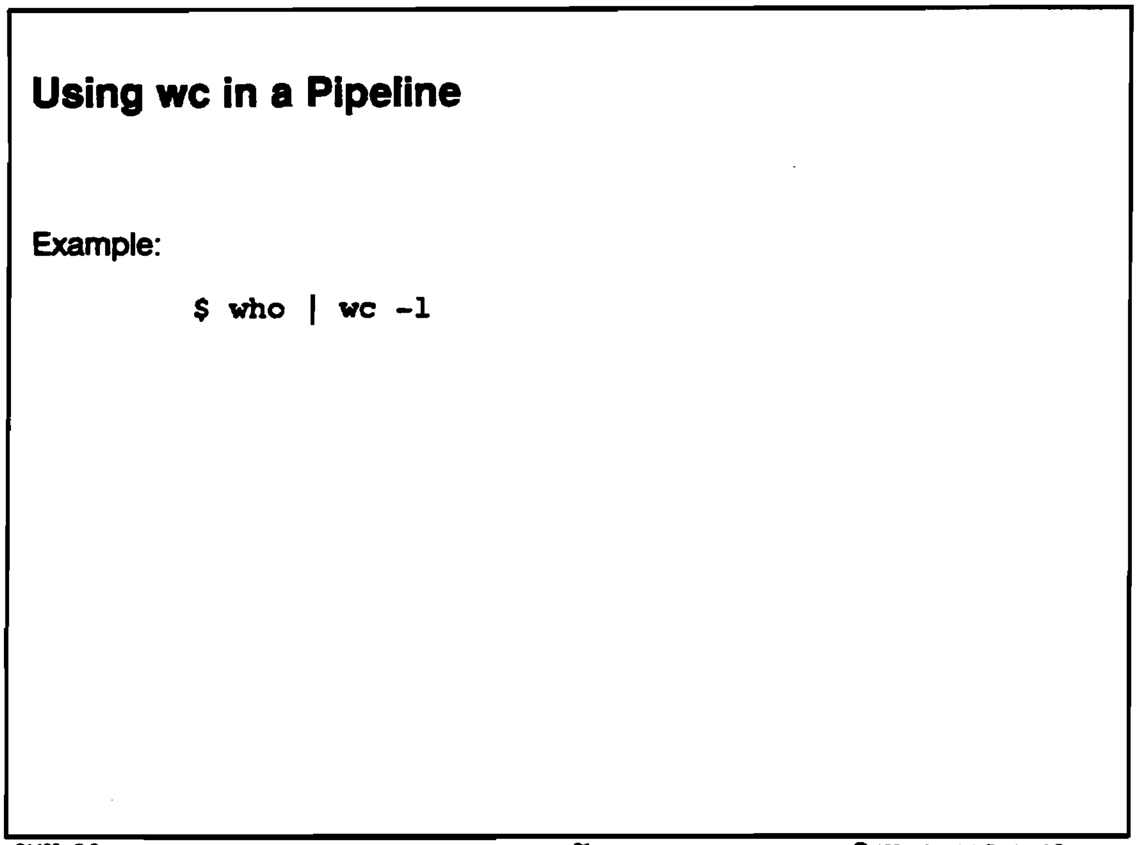
What is the output when more than one file is used as an argument to `wc`?

Answer: The output from each file is listed separately; the last line output is the grand total.

Transition

Here is an example of using `wc` in a pipeline.

9-8. SLIDE: Using wc in a Pipeline



Using wc in a Pipeline

Example:

```
$ who | wc -l
```

51489 9-8. 81 © 1991 Hewlett-Packard Company

Student Notes

The who command reports which users are logged in to the system, the terminal they are using, and when they logged in. What if there were many users logged in, and you did not want to manually count the number of lines output by who? You could redirect the output of who to a file and then use `wc -l` to display how many lines the file contained.

```
$ who > howmany
$ wc -l howmany
     12
```

Again, you could save yourself work and the trouble of creating a temporary file by connecting the output of the who command to the input of the `wc -l` command using a *pipe* (recall that wc reads stdin):

```
$ who | wc -l
     12
```

Now the output of the who command is read by `wc -l`, which will display the result on your terminal.

9-8. SLIDE: Using wc in a Pipeline

Purpose

To give an example of a pipeline.

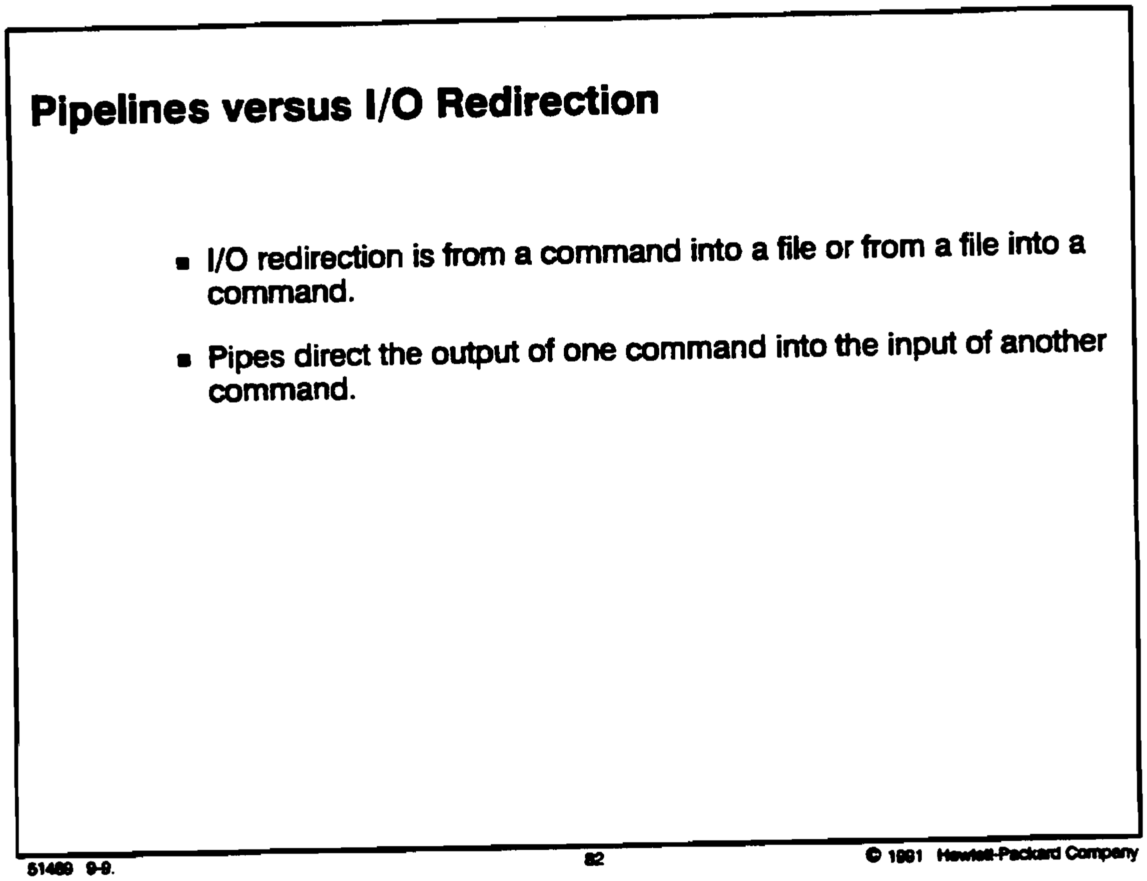
Key Points

who produces output, and wc will read stdin when it is not given a file name argument. This allows them to work well together in a pipeline.

Transition

Pipes and I/O redirection are *not* the same thing!

9-9. SLIDE: Pipelines versus I/O Redirection



Pipelines versus I/O Redirection

- I/O redirection is from a command into a file or from a file into a command.
- Pipes direct the output of one command into the input of another command.

51489 9-9. 82 © 1991 Hewlett-Packard Company

Student Notes

I/O redirection is used to direct output from a command to a file or from a file to a command. Pipes do *not* deal with files. They are used to direct output from one command into the input of another command.

You can use pipes and I/O redirection together, however, to make your command lines even more powerful.

9-9. SLIDE: Pipelines versus I/O Redirection

Purpose

There may be some leftover confusion concerning the similarities and differences between I/O redirection and pipes. This should help.

Key Points

I/O redirection is file to command or command to file. Pipes are command to command.

Teaching Question

What is the obvious difference between using pipelines versus redirection?

Answer: A pipeline is from process to process; redirection is from file to process or process to file.

Transition

Here are some exercises using redirection and pipes.

9-10. LAB: Exercises

1. Create a long listing of your home directory and save it in a file called `myfiles`.

2. Use output redirection to create a file that contains:

- Today's date and time.
- Your current directory.
- A list of who is on the system.

3. Use a pipeline to display a long listing of the `/etc` directory one screen at a time.

4. Construct a pipeline to determine how many files are in your home directory.

5. `sort` is a command that reads stdin, sorts the lines of input, and then writes the sorted lines to stdout. If you give it a file name as an argument, it will sort the lines of the file and write the result on stdout without ever changing the input file.

Use a pipeline to produce a sorted listing of the users on the system.

9-10. LAB: Exercises

1. Create a long listing of your home directory and save it in a file called `myfiles`.

[illegible]

2. Use output redirection to create a file that contains:

- Today's date and time. date > date.file
- Your current directory. pwd >> date.file
- A list of who is on the system. [illegible]

3. Use a pipeline to display a long listing of the `/etc` directory one screen at a time.

[illegible]

4. Construct a pipeline to determine how many files are in your home directory.

[illegible]

5. `sort` is a command that reads stdin, sorts the lines of input, and then writes the sorted lines to stdout. If you give it a file name as an argument, it will sort the lines of the file and write the result on stdout without ever changing the input file.

Use a pipeline to produce a sorted listing of the users on the system.

[illegible]

9-10. LAB: Exercises

Purpose

To show some practical uses of redirection and pipelines.

Presentation Suggestions

Go over the exercises to make sure everyone understands what they are to do. Make sure everyone at least tries the advanced exercise. `sort` is a very useful filter.

At the end of the exercises, give the answers on the board, including those for the advanced exercises.

Transition

The line printer facility is used quite often on UNIX systems. We will take a look at it next. You can use I/O redirection and pipes with the line printer.

Solutions

1. Create a long listing of your home directory and save it in a file called `myfiles`.

Answer:

```
$ cd
$ ls -l > myfiles
$
```

2. Use output redirection to create a file that contains:

- Today's date and time.
- Your current directory.
- A list of who is on the system.

Answer:

```
$ date > file
$ pwd >> file
$ who >> file
```

3. Use a pipeline to display a long listing of the `/etc` directory one screen at a time.

Answer:

```
$ ls -l /etc | more
```

4. Construct a pipeline to determine how many files are in your home directory.

Answer:

```
$ ls | wc -l
     18
```

Advanced:

5. `sort` is a command that reads stdin, sorts the lines of input, and then writes the sorted lines to stdout. If you give it a file name as an argument, it will sort the lines of the file and write the result on stdout without ever changing the input file.

Use a pipeline to produce a sorted listing of the users on the system.

Answer:

```
$ who | sort
chip      tty1p4      Jul 25 06:43
frank     tty1p5      Jul 27 07:42
ralph     tty2p0      Jul 27 09:09
linda     tty3p2      Jul 25 16:07
```

Module 10 — Using the Line Printer

Objectives

Upon completion of this module, you will be able to do the following:

- Explain the purpose of the line printer spooler system.
- Identify and use the line printer spooler commands used to interact with the system.
- Monitor the status of the line printer spooler system.

Module 10 — Using the Line Printer

Overview of Module 10

This module will discuss the line printer spooler system that is available with most UNIX systems. During the course of this module, we will look at the terminology of the line printer system and commands that users can run to interact with the lp system.

Motivation

The management of the line printer spooler system is exclusively an administrative task. Users need to know the lp spooler commands that will allow them to interact with the lp system.

Exercises

The terminal exercises are at the end of the module. The system default printer should be *disabled* before students begin the exercises. The line printer can be enabled when students reach the last exercise.

Exercise Files

`new.jersey`

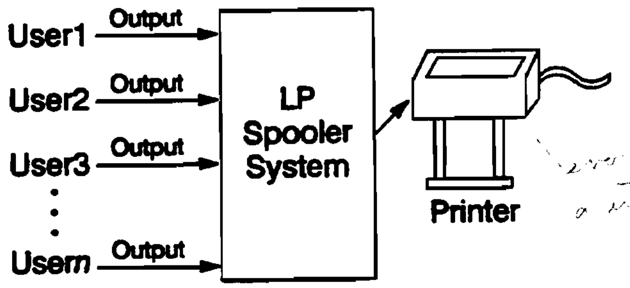
Module Transition

Following the completion of this module we'll take a look the user environment.

10-1. SLIDE: The Line Printer Spooler System

The Line Printer Spooler System

- The lp spooler system is a utility that coordinates printer jobs.
- Allows users to:
  - Queue files to printers.
  - Obtain status of printers and print queues.
  - Cancel any print job.



[illegible]

51480 10-1. 83

[illegible]

Student Notes

The UNIX operating system provides a utility called the **line printer spooler** (or lp spooler) that is used to configure and control printing on your system. The lp spooler is a mechanism that accepts print requests from all of the users on the system and then appropriately configures the printer and prints the requests one at time. Think of the problems we would have if we did not have a spooler. Every time a user wanted to print a file, he or she would have to make sure that no one else was currently printing a file. Two users cannot print to the same printer at the same time

The lp spooler system has many features that allow for smooth running with minimum administrator intervention. You submit your print requests to the lp spooler system, where they will sit in a queue waiting to be printed. You can check which files are queued and the status of the system. You can also cancel a queued printing request if you decide it should not be printed.

[illegible]

10-1. SLIDE: The Line Printer Spooler System

Instructor Notes

Purpose

We start our discussion of the lp spooler system by presenting an overview of some of the things that the spooling system does.

Key Points

If we did not have a spooler to control print jobs it would be up to the users to coordinate who uses the printers and when.

Users can submit requests, obtain information about the spooler, and cancel print requests.

Teaching Question

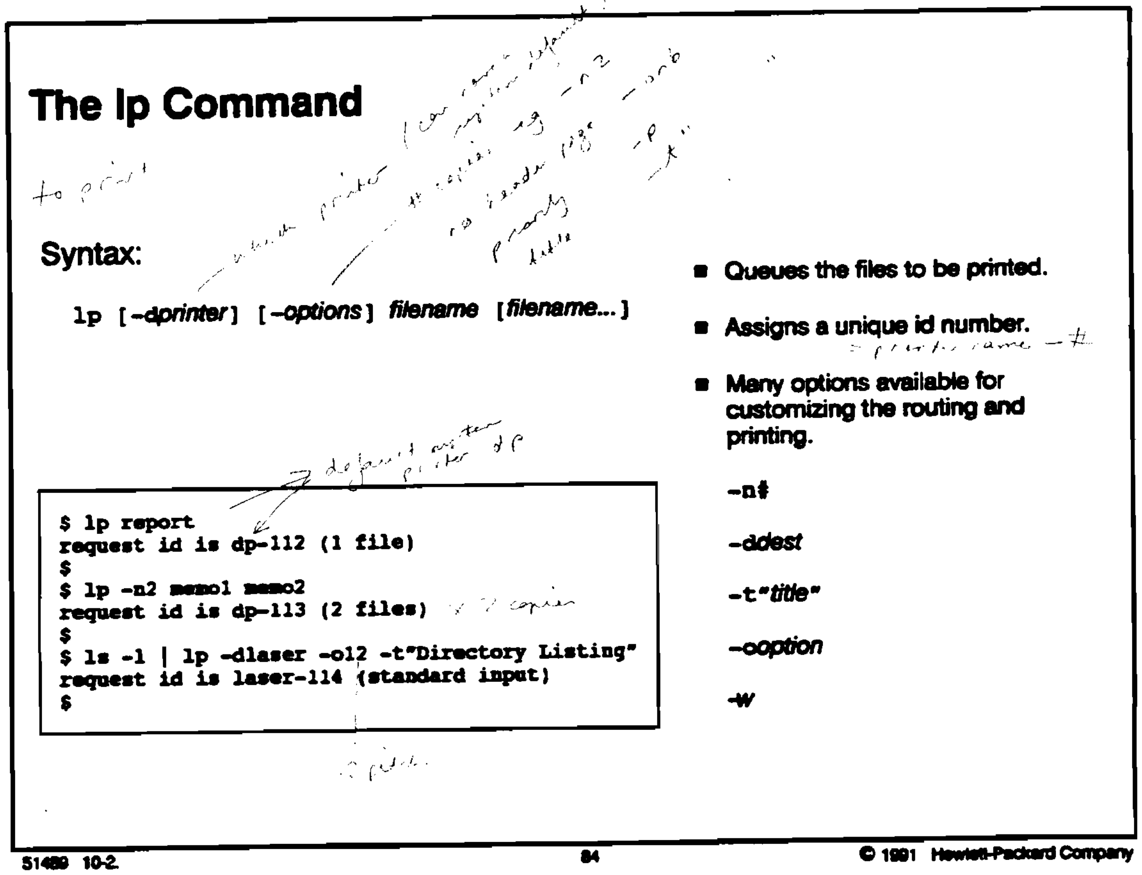
What are some of the things we would want to do through an lp spooling system?

A user can cancel any print request, even if it belongs to another user. Why would you want a user to be able to cancel another user's print requests?

Transition

The `lp` command is probably the most frequently used command of those listed. We will take a look at `lp` first.

10-2. SLIDE: The lp Command



The lp Command

to print [illegible]

Syntax:

```
lp [-dprinter] [-options] filename [filename...]
```

- Queues the files to be printed.
- Assigns a unique id number.

  printer name -#
- Many options available for customizing the routing and printing.

  -n#

  -ddest

  -t"title"

  -ooption

  -w

default printer dp

```
$ lp report
request id is dp-112 (1 file)
$
$ lp -n2 memo1 memo2
request id is dp-113 (2 files)
$
$ ls -l | lp -dlaser -o12 -t"Directory Listing"
request id is laser-114 (standard input)
$
```

copies

pipe

51489 10-2 84 © 1991 Hewlett-Packard Company

Student Notes

The lp command allows the user to queue files for printing. A unique job identification number (called a request id) is given to each request submitted using lp.

lp will queue a file to be printed or it will read standard input.

The simplest use of lp is to give it a file name as an argument and it will queue the file to be printed on the default printer.

The other way lp finds what to print is through standard input. This gives us the ability to use lp with pipes and input redirection. Either way, lp will take standard input and queue it to be printed on the default printer.

The lp command has a number of options available that allow you to customize the routing and printing of your jobs.

The syntax of the lp command is:

```
lp [-ddest] [-nnumber] [-ooption] [-ttitle] [-w] [file ... ]
```

The options to `lp` shown are:

`-n`*number*	Print *number* copies of the request (default is 1).
`-d`*dest*	*dest* is the name of the printer where the request will be printed.
`-t`*title*	Print *title* on the **banner page** of the print out. The banner page is a header page that identifies the owner of the print out.
`-o`*option*	Specify printing options specific to your printer, such as font, pitch, density, raw (for graphics dumps), and so on.
`-w`	Write a message to the user's terminal after the files have been printed.

See lp(1) for a complete listing of available options.

The first example on the slide shows the simplest form of `lp`. We are sending the file `report` to the system default printer. `lp` returns the request id and the number of files submitted to the queue. Here, the file `report` has been sent to printer "dp" and the job is queued with request id dp-112.

In the second example, we are sending `memo1` and `memo2` to be printed and we want 2 copies (`-n2`).

In the third example, we are using a pipe to send the text to `lp`. In this case, `lp` reports a request id of `laser-114` because we specified that the output was to go to the printer named "laser" (`-dlaser`). `lp` also reported that the text has come from standard input. Notice also that the `-o12` argument sets 12 pitch and `-t"Directory Listing"` puts the title "Directory Listing" on the banner page.

In a couple of pages we will see the `lpstat` command which will list the names of all of the printers on the system (and many other things). Using the `-d` option, you can specify which printer to send your request to.

The `-o` option is used to specify other print options that are specific to your model of printer. Ask your system administrator about the other special options available with your printer and `-o`.

Note

`lp` is also a user on the system. If you want to print a file using the `lp` command, you must have the permissions set on the file so the user lp can read it. This means each directory in the path to the file must be readable *and* executable, and the file itself must be readable for lp (normally lp falls under the "other" permissions). This is done so no one can tamper with the requests that are queued

10-2. SLIDE: The lp Command

Purpose

The `lp` command is one of the most frequently executed LP system commands.

Key Points

The `lp` command allows the user to queue files for printing. Requests are known by their request id. This id is assigned to the request when it is submitted using `lp`.

Teaching Question

If there is more than one printer on the system, how can we specify which printer we want `lp` to send a job to?

Answer: Use the -d*printer* option.

Positive Instance

With the first example on the slide, we can assume that every directory that is part of the path to the file has the read and execute bits set for "others." Additionally, we can assume that the file is readable by others.

Evaluation Questions

Why must the directory permissions be set for read and execute by others if you want to print a file?

Why are these permissions less critical when `lp` reads its standard input?

If you want to print a file that has mode 600, how can you submit it for printing without changing permissions on it?

Presentation Suggestions

Explain the syntax of the command and that `lp` can be invoked with a file name or it can read standard input. Have the students explain and try the examples on the slide. Do not forget to disable your printer first!

Chalkboard Notes

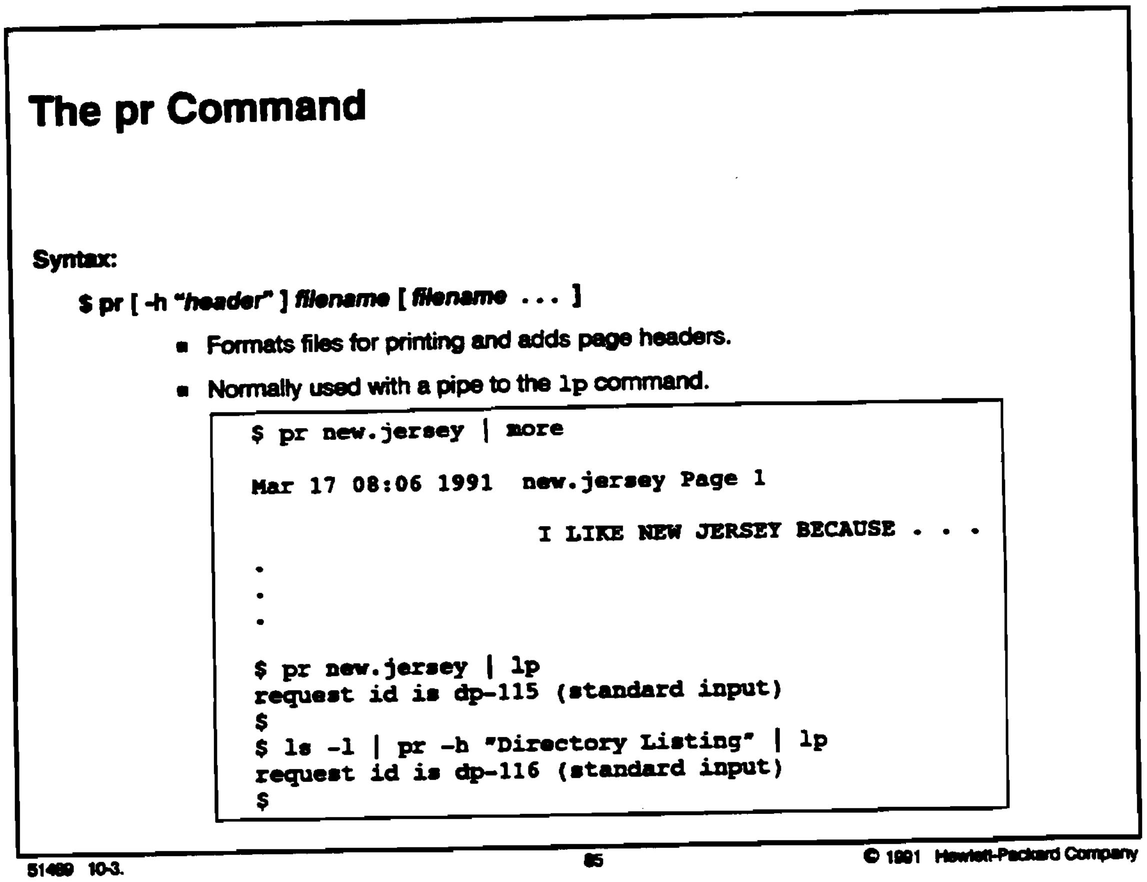
There are many options available to `lp`. It is not our intent to formally discuss all of them here. However, if students have questions about any of the options, an example of some can be drawn on the chalkboard. The lp(1) manual page clearly explains the available options.

You may also want to cover common "-o" options such as `-onb` to suppress printing the banner page, `-o12` to change the print to 12 pitch, `-oraw` to print graphic dumps, and so on.

Transition

`pr` is a handy command to use with the `lp` command.

10-3. SLIDE: The pr Command



The pr Command

Syntax:

$ pr [-h *"header"*] *filename* [*filename* . . .]

- Formats files for printing and adds page headers.
- Normally used with a pipe to the `lp` command.

```
$ pr new.jersey | more

Mar 17 08:06 1991  new.jersey Page 1

                        I LIKE NEW JERSEY BECAUSE . . .
.
.
.

$ pr new.jersey | lp
request id is dp-115 (standard input)
$
$ ls -l | pr -h "Directory Listing" | lp
request id is dp-116 (standard input)
$
```

51499 10-3. 85 © 1991 Hewlett-Packard Company

Student Notes

The pr command is a very handy command used to format output for a printer. It is normally used in conjunction with the `lp` command.

By default, pr will display your file with a header at the top of each page. In this case, a page is 66 lines long (standard printer format). pr will also leave some lines blank at the top and bottom of each page so the output is easier to read. pr writes its output to stdout. This is strange to look at on a terminal because of the page format. That is why pr is normally used with `lp`.

The header includes the date the file was last modified (the same date you see using the `ls -l` command), the file name, and the page number. Using the `-h` option, you can specify your own header. Notice in the third example on the slide, pr is reading the output from the `ls -l` command and piping it to `lp`. In this case, the header contains the current date and time, our header ("Directory Listing"), and the page number.

pr has many more options for formatting text. See pr(1) for more information.

10-3. SLIDE: The pr Command

Instructor Notes

Purpose

To introduce a way to format output for the printer.

Key Points

`pr` has many options. It is probably not a good idea to go into too many options unless the class has a genuine interest.

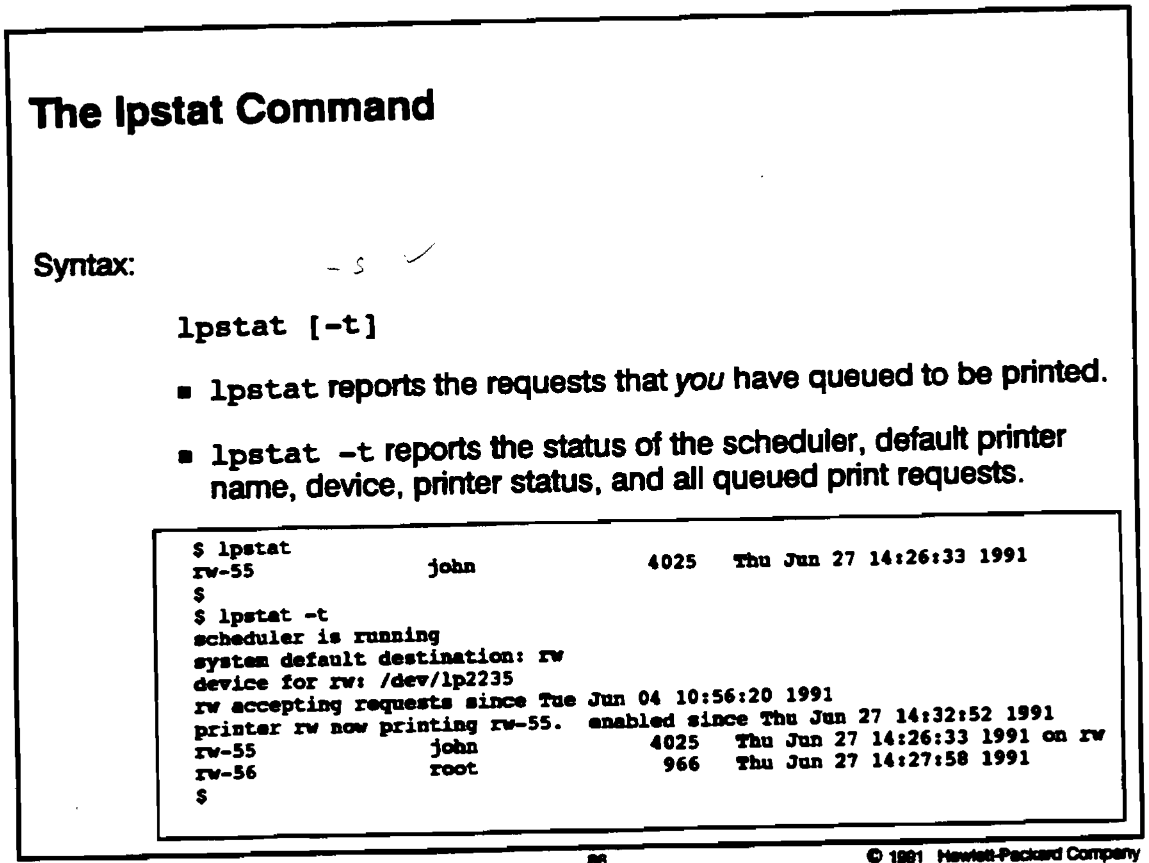
Where Problems Arise

Students may try to use `pr` on their terminals, but it always formats the output to 66 lines unless otherwise specified. This normally puts many blank lines on the screen, and the text instantly scrolls off the top.

Transition

The `lpstat` command can be used by any user. It will print status information concerning the spooler and list information about each printer on the system.

10-4. SLIDE: The lpstat Command



The lpstat Command

Syntax: -s

`lpstat [-t]`

- `lpstat` reports the requests that *you* have queued to be printed.
- `lpstat -t` reports the status of the scheduler, default printer name, device, printer status, and all queued print requests.

```
$ lpstat
rw-55           john            4025   Thu Jun 27 14:26:33 1991
$
$ lpstat -t
scheduler is running
system default destination: rw
device for rw: /dev/lp2235
rw accepting requests since Tue Jun 04 10:56:20 1991
printer rw now printing rw-55.  enabled since Thu Jun 27 14:32:52 1991
rw-55           john            4025   Thu Jun 27 14:26:33 1991 on rw
rw-56           root             966   Thu Jun 27 14:27:58 1991
$
```

51489 10-4. 86 © 1991 Hewlett-Packard Company

Student Notes

The `lpstat` command reports the status of the various parts of the lp spooler system. `lpstat`, when it is used with no options, reports the requests that *you* currently have queued to be printed.

The `-t` option prints all of the status information about all of the printers on the system.

The output of the `lpstat -t` command tells us several things:

`scheduler is running`	The **scheduler** is the program that sends your print requests to the proper printer. Nothing will print if the scheduler is not running.
`system default destination: rw`	`rw` is the name of the default system printer. If you use `lp` without the `-d`*printer* option, your request will be sent to the printer named "rw." Note that your default system printer will probably have a different name (such as "lp").

`device for rw: /dev/lp2235`	This tells the spooler where the printer is connected to the computer.
`rw accepting requests`	This means that the spooler will let you queue files to rw.
`printer rw now printing rw-55`	Request id rw-55 is currently being printed.
`enabled`	Requests can be printed on rw. If a printer is **disabled**, you can submit requests, but they will not be printed until the printer is **enabled** again.

The rest of the lines are the requests to be printed. These fields list the request id, followed by the user making the request, the size of the request, and then the date the request was made.

10-4. SLIDE: The lpstat Command

Purpose

The users should have a way of determining the status of any part of the lp spooler system. The `lpstat` command will report this information.

Key Points

The `lpstat` command reports the status of the various parts of the lp spooler system. It can be used to obtain information about the following:

- The scheduler.
- The printers on the system.
- Printer devices.
- The default system printer.
- Spooler queues.
- Queued requests.

Teaching Question

Could we use `lp` to send a request to a printer that is disabled?

Answer: Yes.

Positive Instance

If a user submits a job to a printer and it does not print immediately, `lpstat` can be used to determine the position of the job on the print queue. If a printer is disabled, it will not print requests.

Evaluation Questions

What does `lpstat` report if you do not give it any options? Answer: It reports the requests that *you* have queued to be printed.

Presentation Suggestions

Explain the purpose of the command and the types of information it will report on. Inform the students that the slide lists only one of the many `lpstat` options. For a complete list of options, they should refer to lpstat(1) in the *HP-UX Reference Manual.*

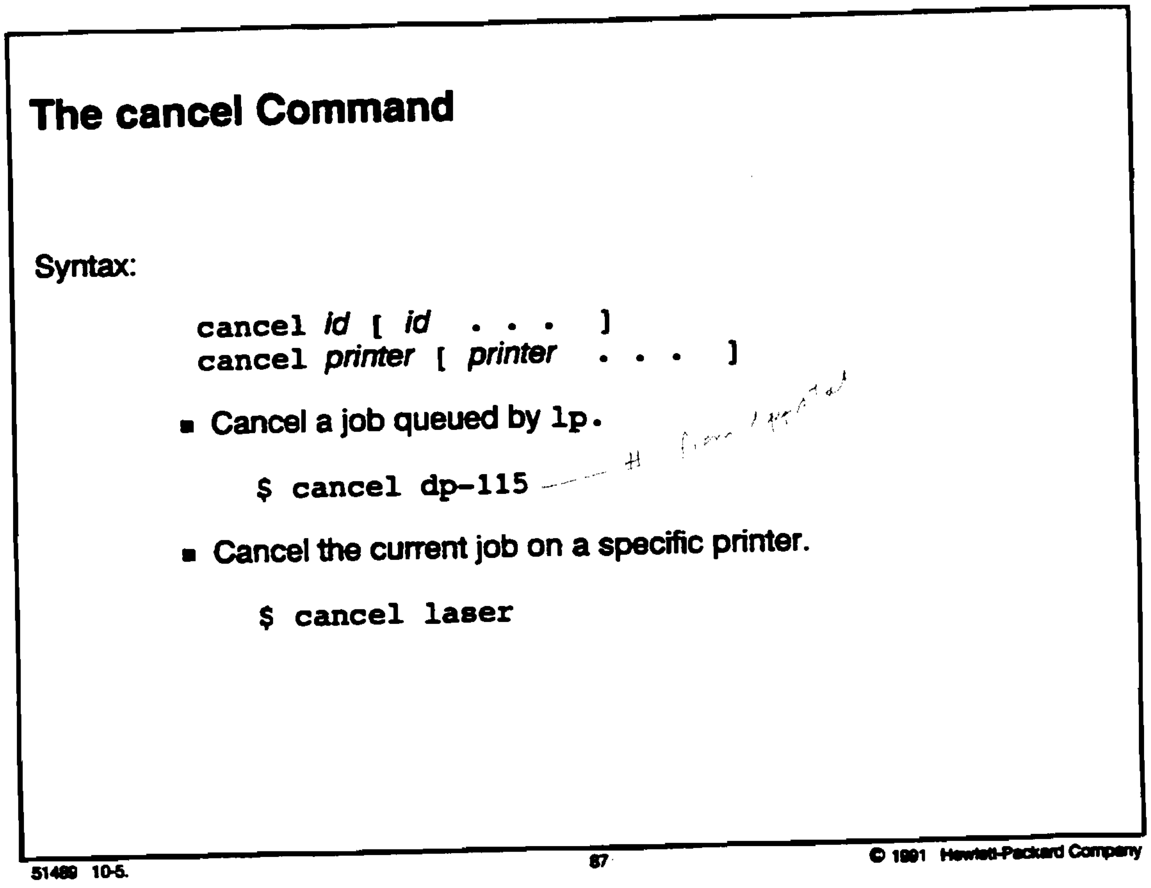
Chalkboard Notes

As each of the options is discussed, draw a sample command line using the option on the chalkboard.

Transition

We will look at the `cancel` command next.

10-5. SLIDE: The cancel Command



The cancel Command

Syntax:

```
cancel id [ id  . . .  ]
cancel printer [ printer  . . .  ]
```

- Cancel a job queued by `lp`.

```
$ cancel dp-115
```

— # from lp started

- Cancel the current job on a specific printer.

```
$ cancel laser
```

51489 10-5. 87 © 1991 Hewlett-Packard Company

Student Notes

The `cancel` command is used to remove requests from the print queue. By canceling the current job on the printer, the next request can be printed. You may want to cancel a request if it is extremely long or if someone tried to print a binary file by mistake (such as `/bin/cat`). Remember, `lp` normally prints text files. Anything else will just "mess up" the printer and waste piles of paper if you do not specify the appropriate options (such as `-oraw` for graphics dumps).

To cancel a request, you must tell the spooler which request to cancel by giving the `cancel` command an argument. Arguments to the `cancel` command can be of two types.

- A request id (as given by `lp` or `lpstat`).
- A printer name.

By giving `cancel` a request id, that specific print request will be canceled. If you give `cancel` a printer name, the current job being printed on that printer will stop and the next request in the queue will start printing.

```
$ lpstat
rw-113               mike            6275   Jul 12 18:46 1989
rw-114               mike            3349   Jul 12 18:48 1989
rw-115               mike            3258   Jul 12 18:49 1989
$ cancel rw-115
request "rw-115" canceled
$ lpstat
rw-113               mike            6275   Jul 12 18:46 1989
rw-114               mike            3349   Jul 12 18:48 1989
$ cancel rw
request "rw-113" canceled
$ lpstat
rw-114               mike            3349   Jul 12 18:48 1989
```

This command can be executed by any user to cancel any request. You can even cancel another user's request; however, mail will be sent to the person whose job was canceled with the name of the user who canceled it. The system administrator can restrict users to canceling only his or her own requests.

10-5. SLIDE: The cancel Command

Instructor Notes

Purpose

Occasionally it may be necessary to remove a printing job from the queue or from the printer (if it is the job currently being printed). The `cancel` command will do this.

Key Points

The `cancel` command is used to remove requests from a destination after the request has been made with `lp`.

By canceling the current job on the printer, the next request will be printed.

This command can be executed by any user on any request.

Teaching Question

Is there any way a specific job can be removed from a print queue, even if it is the job currently printing?

Answer: Use the `cancel` command.

Positive Instance

Line printers cannot print the contents of object files. Nevertheless, some users will mistakenly submit an object file to be printed. The `cancel` command will remove the job from the queue, or, if it has started printing already (undoubtedly sending the printer through all sorts of unnatural gyrations) it will stop the printing shortly.

Negative Instance

The `cancel` command can be used by any user to cancel any other user's job.

Where Problems Arise

The use and purpose of the command should not present any difficulty. Some students may question why a user can cancel another user's job. Give some examples of why you would want to cancel someone's request and it will become clear (for example, it is a nontext file, an extremely large file, and so on).

Evaluation Questions

Why would we ever need to cancel a printing request? Answer: You would not want to print a binary file.

Why is it that any user can invoke `cancel`, even to cancel other users' jobs? Answer: Any user should be able to stop a binary file from being printed.

Can the system administrator keep ordinary users from running `cancel`? Answer: Yes.

Presentation Suggestions

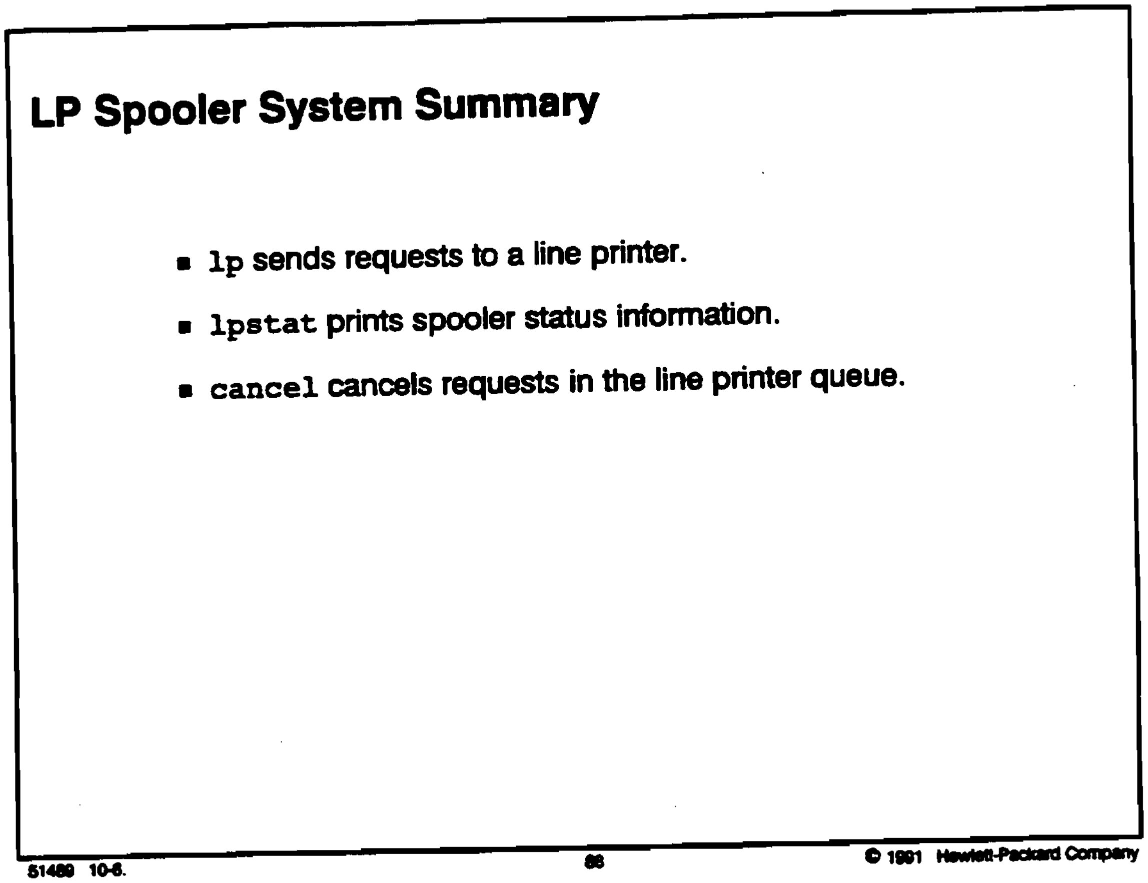
Explain the purpose of the command. Ask the students to provide examples of situations in which this command would be used.

Differentiate between the different arguments that can be passed to `cancel`.

Transition

These are the basic lp spooler commands. Let's summarize what users of the system can do.

10-6. SLIDE: LP Spooler System Summary



LP Spooler System Summary

- `lp` sends requests to a line printer.
- `lpstat` prints spooler status information.
- `cancel` cancels requests in the line printer queue.

51489 10-6. 88 © 1991 Hewlett-Packard Company

Student Notes

The Line Printer Spooler System is a collection of programs that are used to configure and control line printer spooling.

The following commands can be invoked by any user to control the lp spooler system:

`lp`	The `lp` command allows users to queue files to a destination for printing. `lp` can accept what to print in two ways: from a command line argument or from standard input.
`lpstat`	This command reports on the status of the spooling system. This includes whether jobs can be submitted to the queues and the jobs currently waiting to be printed. `lpstat` can also be used to report if the scheduler is running and to identify the default printer name.

`cancel`	The `cancel` command can be used to cancel a specific print job or the current job on a particular printer.

For complete details on each, see the *HP-UX Reference Manual.*

Module 11 — The User Environment

Objectives

Upon completion of this module, you will be able to do the following:

- Describe what happens when someone logs in.
- Describe the user environment variables and their functions.
- Understand and change specific environment variables such as *PATH* and *TERM*.
- Customize the user environment to fit a particular application.

Module 11 — The User Environment

Overview of Module 11

This module describes the basic characteristics of the UNIX user environment. It does not cover all aspects of the environment in detail; however, it does explain it well enough to set some specific environment variables and to intelligently set variables needed by applications.

This module assumes that users are using the Bourne or Korn shells. That is because the Bourne shell is the default shell on HP-UX systems and the Korn shell works in the same way. If your students are running the C shell, you may want to add the necessary extensions. Application users normally do not use the C shell.

The details of shell variables are not covered here. This is meant to be an introduction to setting basic environment variables so the user's application will run correctly.

If the majority of your class already has their applications running or will have their systems set up by an SE, you may want to skip this module.

Motivation

The environment must be set up correctly for applications to run correctly (or at all!). In this module, the student will learn enough about the user environment to do this.

Exercises

The exercises are located at the end of the module.

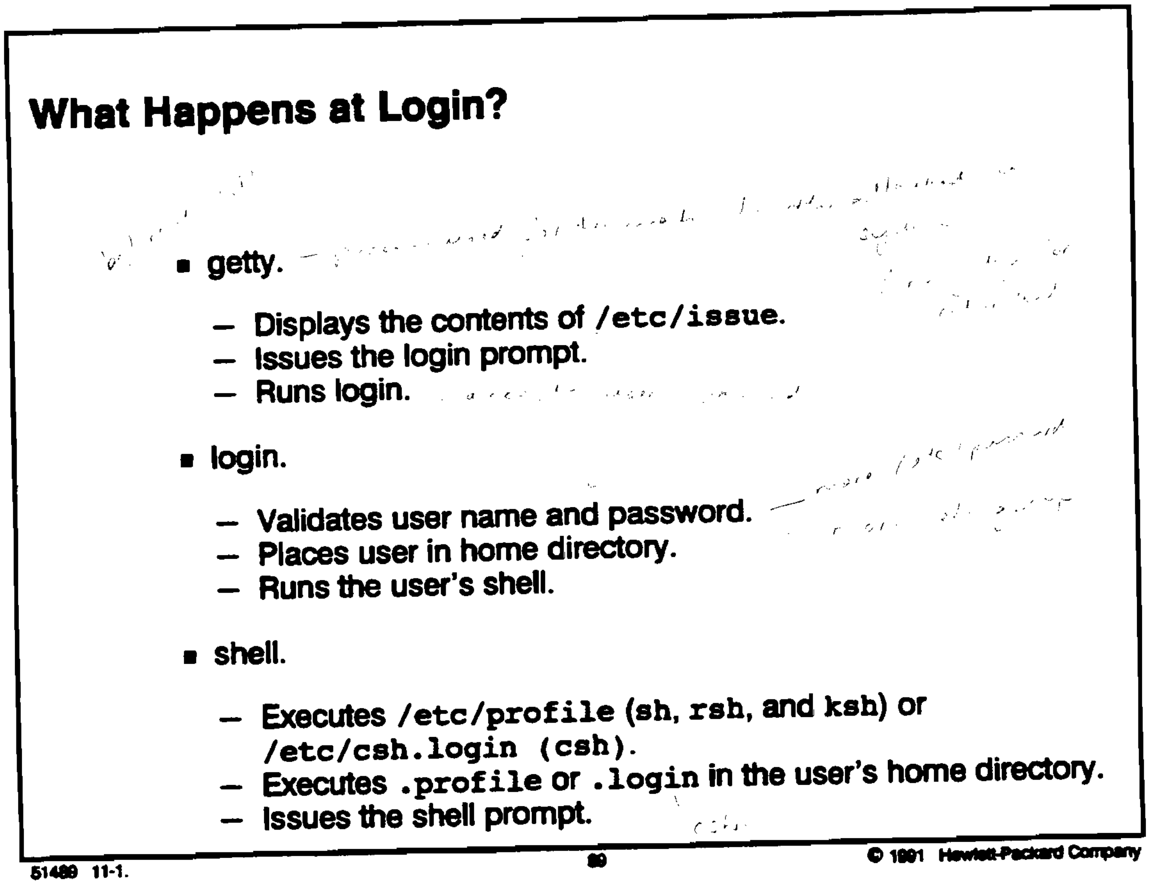
Exercise Files

The user's `.profile`.

Module Transition

The next module gives us some details on how to run your HP-UX system.

11-1. SLIDE: What Happens at Login?



What Happens at Login?

- getty.
  - Displays the contents of `/etc/issue`.
  - Issues the login prompt.
  - Runs login.
- login.
  - Validates user name and password.
  - Places user in home directory.
  - Runs the user's shell.
- shell.
  - Executes `/etc/profile` (`sh`, `rsh`, and `ksh`) or `/etc/csh.login (csh)`.
  - Executes `.profile` or `.login` in the user's home directory.
  - Issues the shell prompt.

51489 11-1. 89 © 1991 Hewlett-Packard Company

Student Notes

When you sit down to do work on the system, you see the `login:` prompt on the screen. When you type your user name, the system reads your name and prompts you for a password. After you enter your password, the system checks your user name and password in the system password file (`/etc/passwd`). If the user name and password you entered are valid, the system will place you in your home directory and start the shell for you. We have seen this happen each time we logged in. Our question is: What really happens when the shell is started?

Once the shell starts running, it will read commands from a system command file called `/etc/profile`. Whenever someone logs in and starts a shell, this file will be read. There is also a file called `.profile` in your home directory. After `/etc/profile` is read, the shell reads your own `.profile`. These two shell programs are used to customize a user's environment.

`/etc/profile` sets up the basic environment used by everyone on the system and `.profile` further tailors that environment to your specific needs. Since everyone uses `/etc/profile`, the system administrator will take care of it. It is your responsibility, however, to maintain you own `.profile` to set up your user environment.

When these two programs are finished, the shell issues the first shell prompt.

The rest of this module describes the user environment and the kinds of things you can do to make your environment fit your applications.

11-1. SLIDE: What Happens at Login?

Purpose

To introduce how the system sets up the initial user environment.

Key Points

Everyone who logs in reads `/etc/profile` when his or her shell is first started. Each user has a `.profile` to further customize his or her environment beyond what `/etc/profile` does.

Evaluation Questions

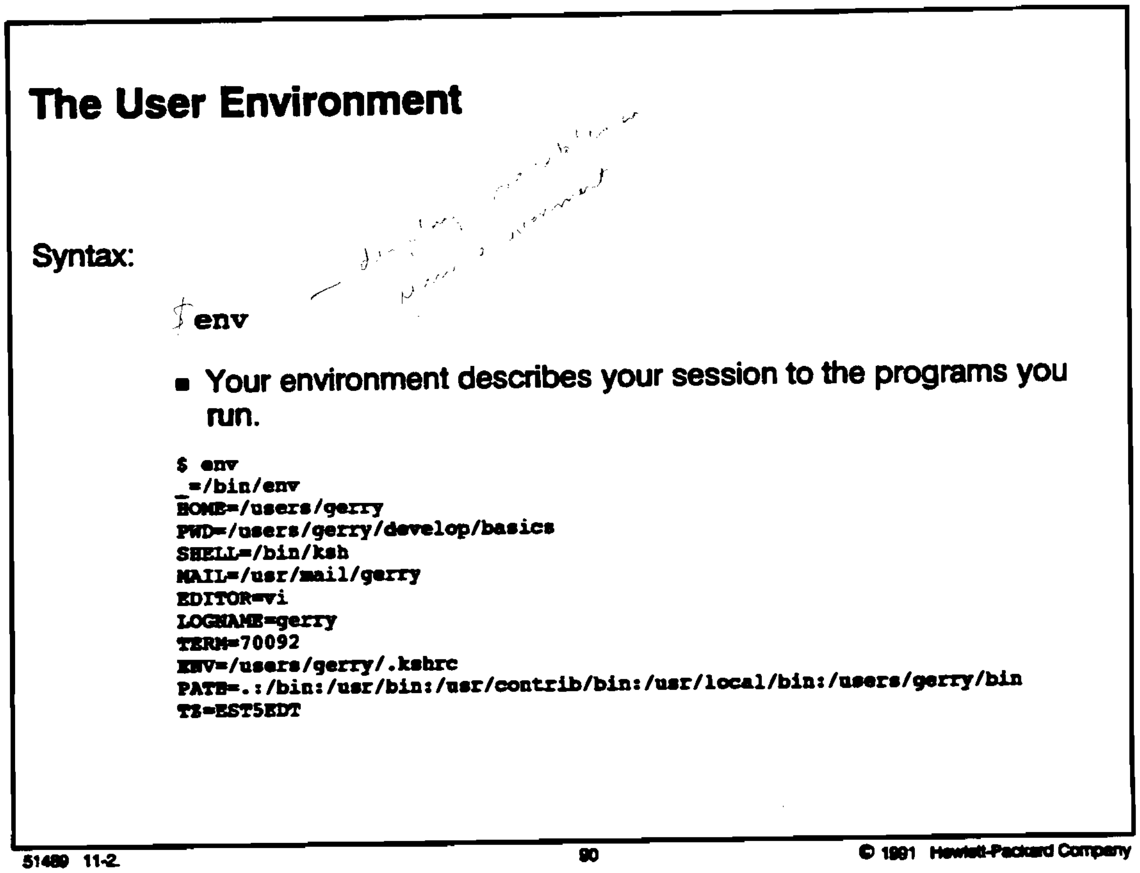
Which users use `/etc/profile`? Answer: Those who log in at the Bourne Shell or Korn Shell.

When is `.profile` read? Answer: When a user enters a Bourne Shell or Korn Shell.

Transition

Let's take a more detailed look at the user environment.

11-2. SLIDE: The User Environment



The User Environment

[illegible]

Syntax:

$env

- Your environment describes your session to the programs you run.

```
$ env
_=/bin/env
HOME=/users/gerry
PWD=/users/gerry/develop/basics
SHELL=/bin/ksh
MAIL=/usr/mail/gerry
EDITOR=vi
LOGNAME=gerry
TERM=70092
ENV=/users/gerry/.kshrc
PATH=.:/bin:/usr/bin:/usr/contrib/bin:/usr/local/bin:/users/gerry/bin
TZ=EST5EDT
```

51489 11-2 90 © 1991 Hewlett-Packard Company

Student Notes

Your environment describes many things about your session to the programs that you run. It describes your session to the system. Your environment contains information concerning the following:

- The path name to your home directory. HOME
- Where to send your electronic mail.
- The time zone you are working in. TZ
- Who you logged in as. LOGNAME
- Where your shell will search for commands. PATH
- Your terminal type and size. TERM
- Other things your applications may need.

export
→ place variable in env

For example, the commands `vi` and `more` need to know what kind of terminal you are using so they can format the output correctly.

An analogy to your user environment is your office environment. In the office, characteristics such as lighting, noise, and temperature are the same for all workers. The factors in your office that are unique to you make up your specific environment. Theses factors include what tasks you are performing, the physical layout of your desk, and how you relate to other people in the office. Your work environment is unique to you just like your user environment is unique.

Many applications require you to customize your environment in some way. This is done by modifying either `/etc/profile` (like changing the lighting in the office), `.profile` (putting a light on your desk), or both. We will only concern ourselves with `.profile` in this module (the system administrator takes care of `/etc/profile` for us).

When you log in, you can check your environment by running the **env** command. It will display every characteristic that you have set in your environment.

In the **env** listing, the words to the left of the "=" are the names of the different **environment variables** that you have set. Everything to the right of the "=" is the value associated with each variable. See env(1) for more details.

Each one of these environment variables is set for a reason. Here are a few common environment variables and their meanings:

TERM, *COLUMNS*, and *LINES*	Describe the terminal you are using.
HOME	Path name to your home directory.
PATH	List of places to find commands.
LOGNAME	User name you used to log in.
ENV and *HISTFILE*	Special Korn shell variables.
DISPLAY	Special X Window variable.

Some of these variables are set for you by the system, while others are set in `/etc/profile` or `.profile`.

11-2. SLIDE: The User Environment

Purpose

Show a sample user environment using `env` and describe the purpose of some of the more important variables.

Key Points

`env` displays all environment variables set by the system and the two login files.

Teaching Question

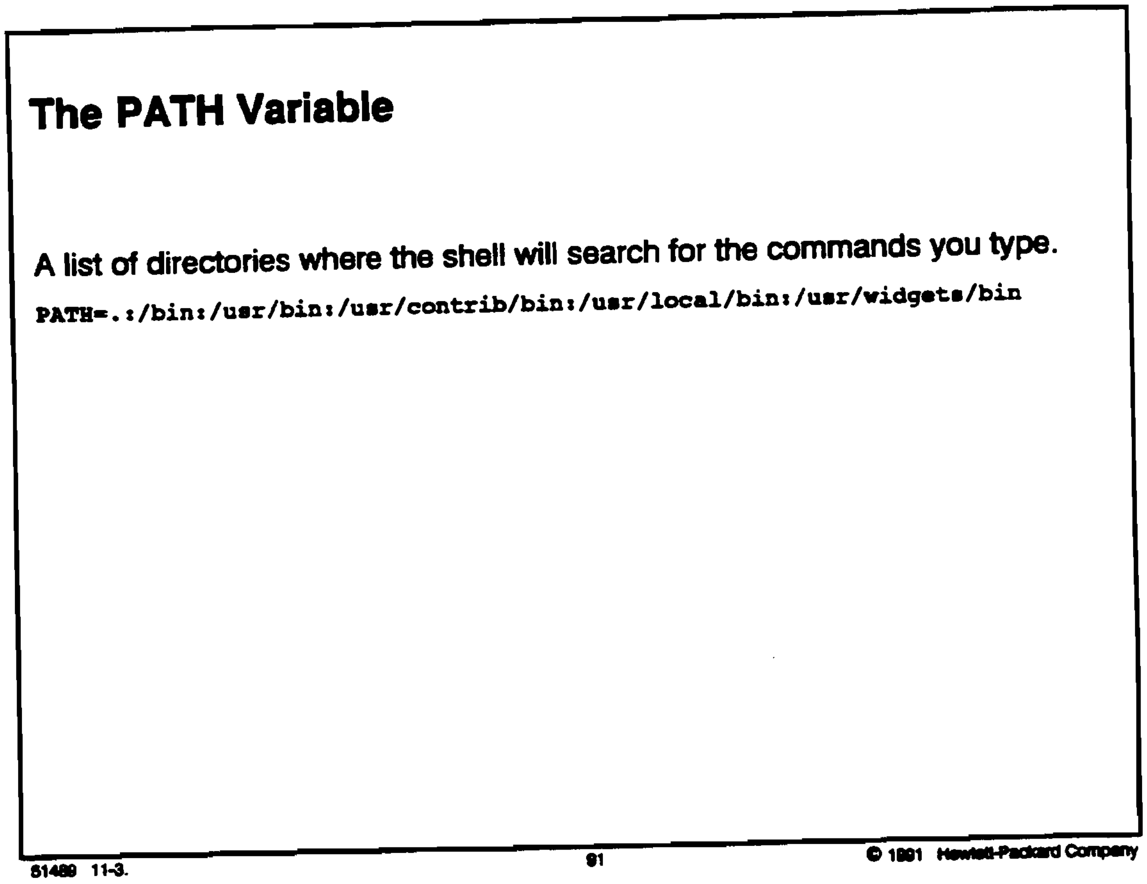
Ask what the other environment variables that are listed might do.

Transition

There are three especially important items in your environment that may have to be customized:

- The *PATH* variable (where to find commands).
- The *TERM* variable (your terminal type).
- The special environment variables for applications.

11-3. SLIDE: The PATH Variable



The PATH Variable

A list of directories where the shell will search for the commands you type.

```
PATH=.:/bin:/usr/bin:/usr/contrib/bin:/usr/local/bin:/usr/widgets/bin
```

61489 11-3. 91 © 1991 Hewlett-Packard Company

Student Notes

The *PATH* variable is a list of directories that the shell will search through to find commands. It gives us the ability to type just a command name instead of the full path name to that command (for example, `vi` instead of `/usr/bin/vi`). This is an example of the default *PATH* variable:

```
PATH=.:/bin:/usr/bin:/usr/contrib/bin:/usr/local/bin
```

This means that when you type a command, the shell will search for that command in the current directory (.). If it does not find the command there, it will look in `/bin`, then `/usr/bin`, and so on until it either finds the command or it runs out of directories to look in. If the command you are trying to run cannot be found in any of the *PATH* directories, you will get the `command: not found` error message on your screen.

For example, let's say that you have a "Widget" application on your system. All of the commands for the application are stored in the directory `/usr/widgets/bin`. If you want to use these commands, you should make sure that this directory is a part of your *PATH*. Then, if you wanted to run the `makewidget` program in this directory, you would simply type `makewidget` on the command line. If you did not have

/usr/widgets/bin in your *PATH*, you would have to type /usr/widgets/bin/makewidget. The benefits of having your *PATH* variable set correctly are obvious.

To add this widgets directory to your *PATH*, type the following command line at the shell prompt:

```
$ PATH=$PATH:/usr/widgets/bin
```

This command appends the new directory (/usr/widgets/bin) to your current *PATH* (*$PATH*) and then assigns it as the new *PATH* variable.

If you want this change made every time you log in, simply add this command to your .profile after the *PATH* assignment that is already there.

If all users on your system run Widgets, then your administrator could add /usr/widgets/bin to the default *PATH* in /etc/profile instead.

11-3. SLIDE: The PATH Variable

Purpose

To describe the importance of the *PATH* variable and how it can be changed.

Key Points

The `command: not found` error normally occurs because an improper command name was entered or the *PATH* was not set correctly.

Teaching Question

If I had a directory named `bin` under my home directory, how could I add it to my *PATH*? `PATH=$PATH:/users/gerry/bin` works; however, the sharp students may realize that `PATH=$PATH:$HOME/bin` also works.

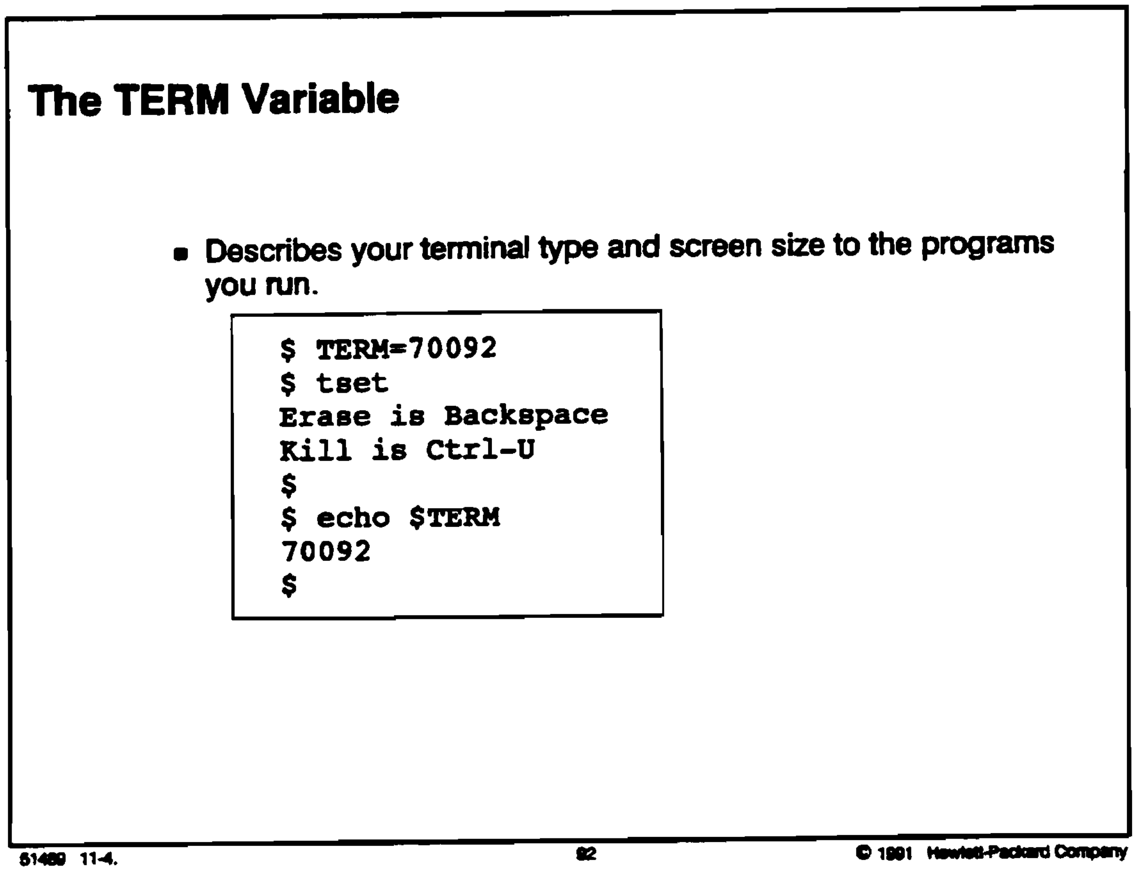
Evaluation Question

What would happen if we typed `makewidget`, but our *PATH* did not include `/usr/widgets/bin`? Answer: We would get `ksh: makewidget: not found.`

Transition

Another important variable is *TERM*.

11-4. SLIDE: The TERM Variable



The TERM Variable

- Describes your terminal type and screen size to the programs you run.

```
$ TERM=70092
$ tset
Erase is Backspace
Kill is Ctrl-U
$
$ echo $TERM
70092
$
```

51489 11-4. 82 © 1991 Hewlett-Packard Company

Student Notes

TERM is the environment variable that describes the type of terminal you have. For many commands to run correctly, they need to know what kind of terminal you are using. For example, the `ls` command needs to know how many columns there are on the screen, `more` needs to know how many lines there are, and `vi` needs to know both how many columns and how many lines there are plus much more information about your terminal type in order to work properly.

The default `.profile` sets up the terminal by prompting the user for the proper terminal type in the following fashion:

```
TERM=(hp)
```

At this prompt, you can either press (Return) to set the terminal type to"hp" or you can type the name of the terminal you are using. Terminal type "hp" is a standard 80 column by 24 line Hewlett-Packard terminal. The table below specifies different types of terminals and how the terminal type should be set. The terminal type is set using the terminal's model number (such as 2392, 70092, and so on).

Terminal Type	Setting for Terminal Type
Standard 80 column by 24 line Hewlett-Packard Terminal	hp
Alphanumeric Display Terminal	70092 or 2392
Medium resolution graphics display (512 x 600 pixels)	3001 or hp3001
High resolution graphics display (1024 x 768 pixels)	300h or hp300h
HP 98550 display station (1280 x 1024 pixels)	98550,hp98550, 98550a, or hp98550a
HP 98720 or HP 98721 SRX (1280 x 1024 pixels)	98720,hp98720, 98720a, hp98720a, 98721, hp98721, 98721a, or hp98721a
HP 98730 or HP 98731 Turbo SRX (1280 x 1024 pixels)	98730, hp98730, 98730a, hp98730a, 98731, hp98731, 98731a or hp98731a

Your administrator may have set up your system so it never asks you about your terminal type. In this case you should check the *TERM* variable using the `env` command. If you are using a workstation with only one display, the *TERM* variable is probably set correctly and should not have to be changed.

If your *TERM* variable is not set correctly, you can change it with the following command:

```
$ TERM=type
```

where *type* is the terminal type listed in the table. Of course, you could add this same line to your `.profile` and the change would be made every time you log in.

The *LINES* and *COLUMNS* variables also help describe your terminal when you are in special applications such as a windowing system in which the window size may be larger or smaller than the defaults set by the *TERM* type.

If your terminal is acting strangely when you are using commands such as `more` and `vi`, check the *TERM* variable. If it is set correctly, execute the `tset` command. This will reset the terminal characteristics using the terminal type found in the *TERM* variable.

11-4. SLIDE: The TERM Variable

Instructor Notes

Purpose

The *TERM* is quite often incorrect unless the system administrator has set it up already. Users need to know what the correct terminal types are for their systems.

Key Points

If you are on a workstation or PC and the *TERM* variable is correct, you will probably never have to change it.

The command eval ‘tset -s *type*‘ will set the terminal type in one step.

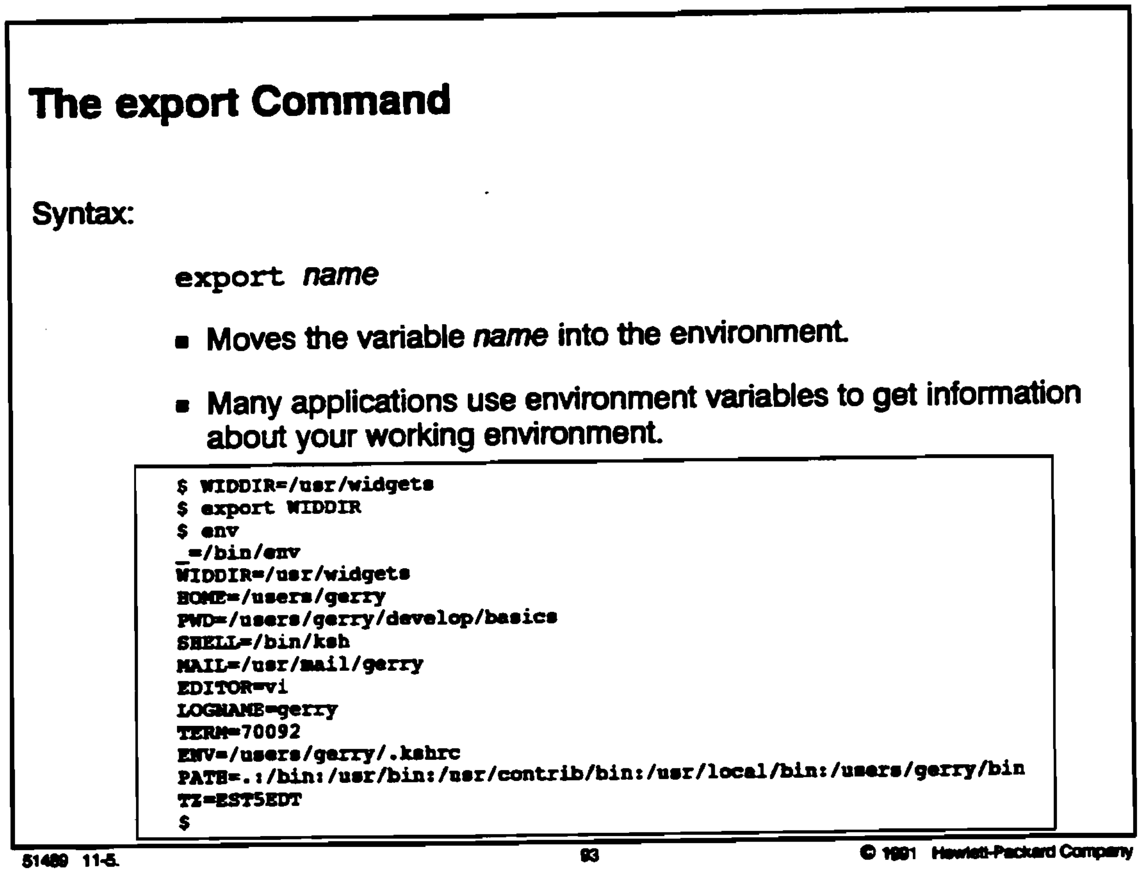
Evaluation Question

What would the more command do if *LINES* was set to 30 but you were using a 24-line display? Answer: The more command would give you 30 lines, so 6 lines at the top would scroll off the display.

Transition

Other variables may affect how your applications run.

11-5. SLIDE: The export Command



The export Command

Syntax:

`export` *name*

- Moves the variable *name* into the environment.
- Many applications use environment variables to get information about your working environment.

```
$ WIDDIR=/usr/widgets
$ export WIDDIR
$ env
_=/bin/env
WIDDIR=/usr/widgets
HOME=/users/gerry
PWD=/users/gerry/develop/basics
SHELL=/bin/ksh
MAIL=/usr/mail/gerry
EDITOR=vi
LOGNAME=gerry
TERM=70092
ENV=/users/gerry/.kshrc
PATH=.:/bin:/usr/bin:/usr/contrib/bin:/usr/local/bin:/users/gerry/bin
TZ=EST5EDT
$
```

51489 11-5. 93 © 1991 Hewlett-Packard Company

Student Notes

Many applications require that you have some environment variables set to tell them where to find files or programs, or maybe what options and arguments to use.

Let's look at our "Widgets" application again. The installation manual says that it requires that a variable called *WIDDIR* be set to the directory `/usr/widgets`. You will have to execute these commands if you want to run the Widgets application:

```
$ WIDDIR=/usr/widgets
$ export WIDDIR
```

Note that there are no spaces around the "=" in the assignment command.

The first line sets the value for *WIDDIR* and the second line (`export WIDDIR`) places *WIDDIR* into the environment. We did not have to `export` *PATH* or *TERM* because they were already exported by the system when `/etc/profile` was run.

You should add these two lines to your `.profile` if you want *WIDDIR* set in the environment every time you login.

11-5. SLIDE: The export Command

Instructor Notes

Purpose

To describe how to set specific environment variables that applications require.

Key Points

Make sure that students know that there are no spaces around the "="s in the assignments commands.

Almost every application uses some number of environment variables. The applications will not operate correctly if the variables are not set.

Make sure the students understand the use of the `export` command and why we used it here and not with *PATH* or *TERM*.

Presentation Suggestions

If you have an advanced group, you might go into some detail about the other environment variables that students may see in an `env` listing and describe their purpose. For example, the Korn shell uses the variables *ENV*, *HISTFILE*, and *HISTSIZE*; X11 uses *DISPLAY*, *WMDIR*, and *WINDOWID*, and so on.

Transition

Here are some exercises in setting and using environment variables.

11-6. LAB: Exercises

1. Modify your `.profile` to include these changes:

- Add `/usr/widgets/bin` to the *PATH*.
- Add the `bin` directory under your home directory to the *PATH*.
- Create a variable named *WIDDIR* and set its value to `/usr/widgets`.

Advanced:

How can you add the `bin` directory to the *PATH* without typing the full path name to your home directory on the line?

2. Have the new variables from the last exercise changed in your environment yet? Do whatever is necessary to make these changes active. Use the `env` command to make sure the changes were made correctly.

3. Temporarily set your terminal type to "unknown" (Do not put this change in the `.profile`.) Use `env` to make sure it is set correctly.

Try to edit the file `new.jersey` using `vi`. What happened?

4. Exit `vi` and set your *TERM* variable to normal without logging out.

Use the `tset` command to reset the terminal characteristics.

5. *PS1* is a shell variable that is used as your shell prompt. Change the value of *PS1* so your prompt is `What now?` .

11-6. LAB: Exercises

Purpose

To show the need for having the correct terminal type and path set.

Transition

Some optional modules follow. Use all of the modules that apply to the students' needs.

Solutions

1. Modify your `.profile` to include these changes:

- Add `/usr/widgets/bin` to the *PATH*.
- Add the `bin` directory under your home directory to the *PATH*.
- Create a variable named *WIDDIR* and set its value to `/usr/widgets`.

Advanced:

How can you add the `bin` directory to the *PATH* without typing the full path name to your home directory on the line?

Answer:

The modified `.profile` should have lines that look something like this:

```
# Set up the search paths:
PATH=.:/bin:/usr/bin:/usr/contrib/bin:/usr/local/bin
PATH=$PATH:/usr/widgets/bin:$HOME/bin

# Set up the shell variable:

WIDDIR=/usr/widgets
export WIDDIR
```

2. Have the new variables from the last exercise changed in your environment yet? Do whatever is necessary to make these changes active. Use the `env` command to make sure the changes were made correctly.

Answer:

No, the variables in `.profile` are only set when you log in to the system. Therefore, you must log out and log back in. The `env` output should look something like this:

```
_=/bin/env
```

```
WIDDIR=/usr/widgets
HOME=/users/gerry
PWD=/users/gerry
SHELL=/bin/ksh
MAIL=/usr/mail/gerry
EDITOR=vi
LOGNAME=gerry
TERM=70092
ENV=/users/gerry/.kshrc
PATH=.:/bin:/usr/bin:/usr/contrib/bin:/usr/local/bin:/usr/widgets/bin
TZ=EST5EDT
```

3. Temporarily set your terminal type to "unknown" (Do not put this change in the `.profile`.) Use `env` to make sure it is set correctly.

Try to edit the file `new.jersey` using `vi`. What happened?

Answer:

To set *TERM*, use the command `TERM=unknown`.

When you try to run `vi`, it enters a line mode called **open mode** because the terminal type is not set correctly. The `vi` commands still work in open mode, but you only see one line at a time instead of a full screen.

4. Exit `vi` and set your *TERM* variable to normal without logging out.

Use the `tset` command to reset the terminal characteristics.

Answer:

To set the terminal type back, type `TERM=`*type*, where *type* is the terminal type listed in the terminal type table shown earlier.

Advanced Exercise:

5. *PS1* is a shell variable that is used as your shell prompt. Change the value of *PS1* so your prompt is `What now?` .

Answer:

```
$ PS1="What now? "
What now?  whoami
gerry
What now?
```

Appendix A — Running Your HP-UX System

Objectives

Upon completion of this module, you will be able to do the following:

- Describe the system boot procedure.
- Check what is running on the system.
- Kill unwanted processes.
- Describe the system shutdown guidelines.
- Cleanly shut down the system using the `shutdown` command.

Appendix A — Running Your HP-UX System

Overview of Module A

This module looks at how an HP-UX system is booted and what happens during the boot operation. Two ways of booting the system will be presented. This module also looks at procedures for shutting down an HP-UX computer.

Motivation

The individual responsible for booting the system is normally the system administrator. There will be occasions for a user to either start the system or shut it down. In a workstation environment, the typical user can also be "root." That is why these topics are presented in this course.

Consequently, users should be familiar with the different ways the system can be booted and what happens during the boot operation. They should also know how to properly shut the system down.

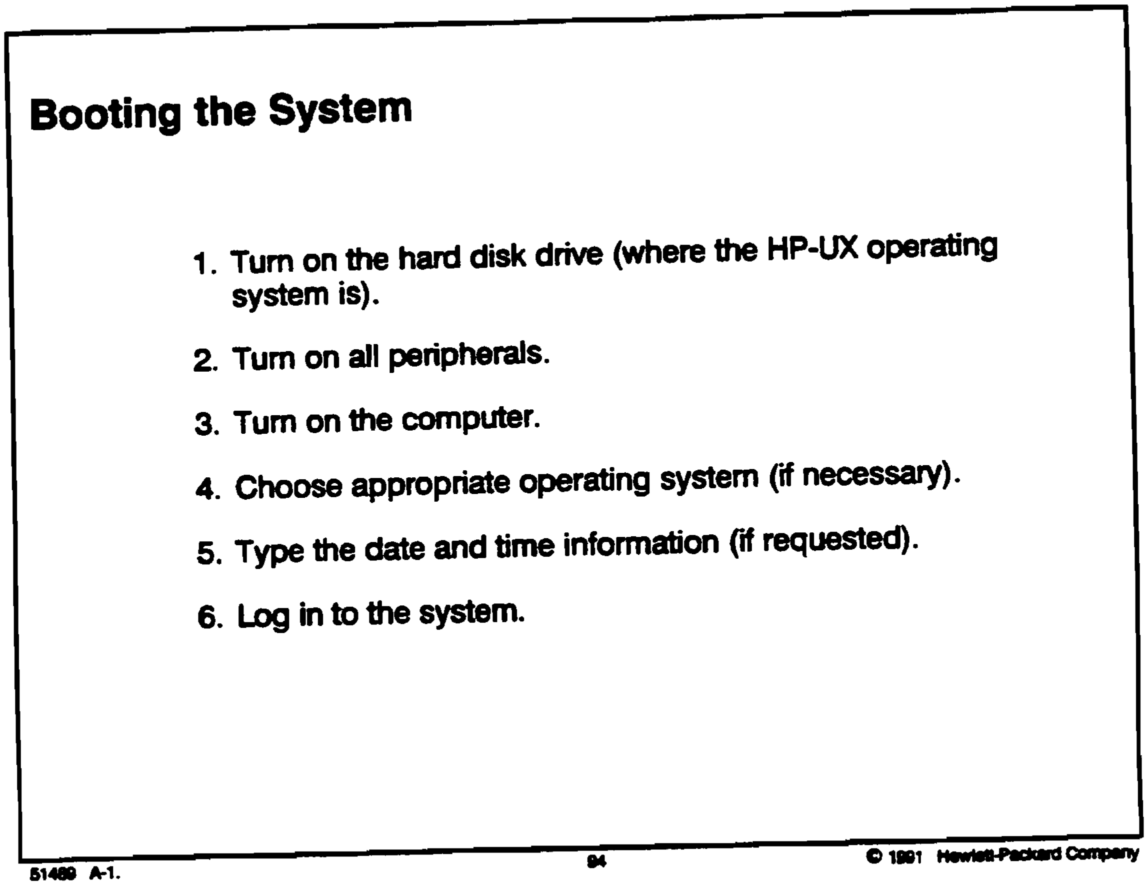
Exercises

The exercises for this module are found at the end of the module.

Module Transition

The rest of the modules in the class are optional. Cover them as needed to meet the students' interests.

A-1. SLIDE: Booting the System



Booting the System

1. Turn on the hard disk drive (where the HP-UX operating system is).
2. Turn on all peripherals.
3. Turn on the computer.
4. Choose appropriate operating system (if necessary).
5. Type the date and time information (if requested).
6. Log in to the system.

51489 A-1. 94 © 1991 Hewlett-Packard Company

Student Notes

Booting the system is bringing your computer from a powered down state to a functional state running an operating system so you can log in and do work. Anyone can boot the system by following these steps:

1. Turn on power to the hard disk and the tape drive and wait for them to complete their self tests.
2. Turn on power to all other peripherals such as the display, printers, and plotters.
3. Turn on power to your computer.

If you leave the system alone, it will boot the first operating system it finds. This usually boots HP-UX from a hard disk or diskless server. This is called an **unattended boot**.

There are ways to interrupt the boot process so that we can interactively boot the system. This is called an **attended boot** and is normally used only by system administrators when they want to boot an operating system other than HP-UX. Attended boots must be performed from the system console. The console is a special display where HP-UX will send all of its bootup, shutdown, and error messages and where the system administrator can conduct administrative activities.

A-1. SLIDE: Booting the System

Purpose

To introduce the proper method of booting up a computer. There will be times when the application user will need to power up the computer so he or she can log in and do work.

Key Points

Point out that the students *do not* need to boot in attended mode if they have no other bootable systems installed. In this class, few people will have to worry about attended mode boots. The exception would be if they had two operating systems such as HP-UX and Rocky Mountain BASIC (RM BASIC) on the same disk. They would have to perform an attended mode boot if they wanted to boot RM BASIC.

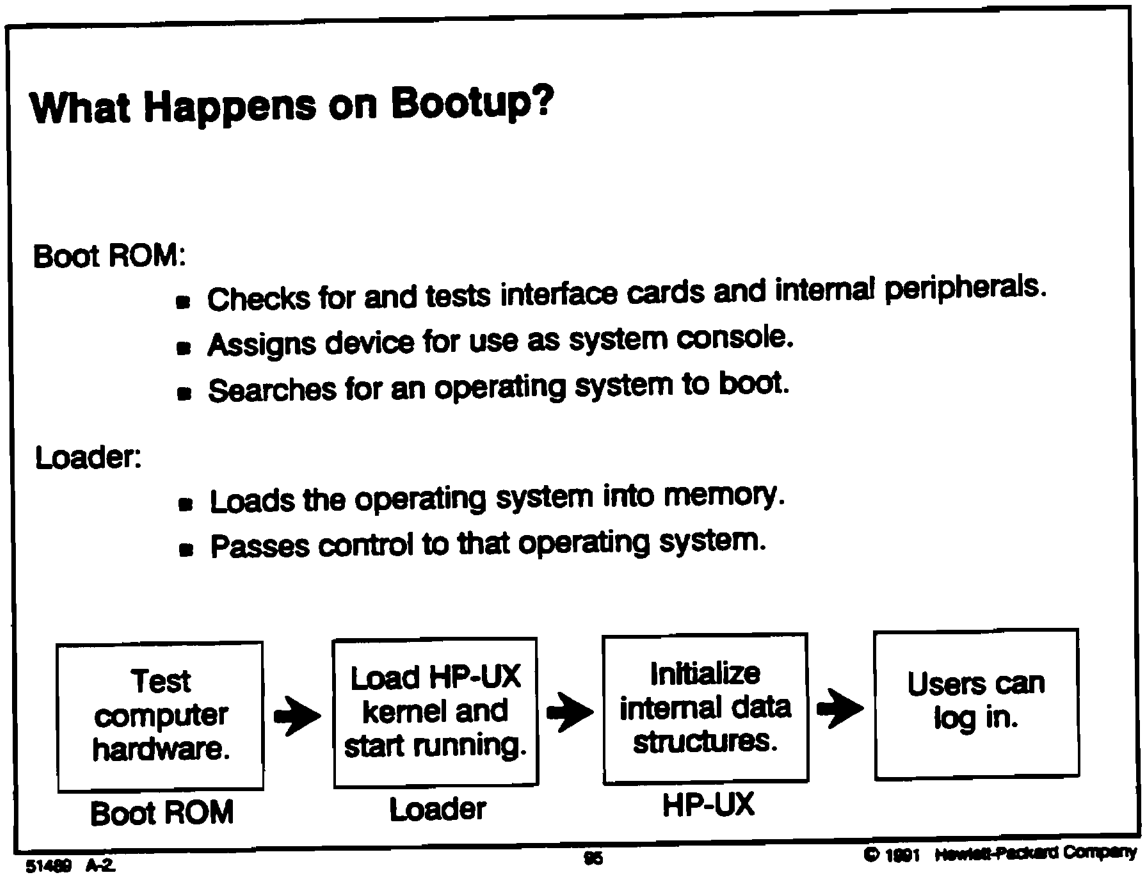
Evaluation Question

What would happen if you powered on the computer before you powered on your display or disk? Answer: The computer would not be able to find an operating system.

Transition

Let's take a look at what happens when you boot HP-UX.

A-2. SLIDE: What Happens on Bootup?



What Happens on Bootup?

Boot ROM:

- Checks for and tests interface cards and internal peripherals.
- Assigns device for use as system console.
- Searches for an operating system to boot.

Loader:

- Loads the operating system into memory.
- Passes control to that operating system.

Boot ROM	Loader	HP-UX	
Test computer hardware. →	Load HP-UX kernel and start running. →	Initialize internal data structures. →	Users can log in.

51489 A-2 95 © 1991 Hewlett-Packard Company

Student Notes

From the time you power on your computer until you have successfully logged in, many tasks are automatically performed by the system. These tasks include the following:

- Testing the computer hardware.
- Locating and loading an operating system.
- Initializing system resources.

The **Boot ROM** is a set of computer chips programmed to control the system bootup. It is the Boot ROM's job to test the computer hardware (including memory and interface cards) and to locate all bootable operating systems that may be available to the system. The Boot ROM will start the **secondary loader** to load the first operating system it finds into memory. If you are doing an attended boot, it will load the operating system you choose into memory.

Once the operating system is loaded, it begins to execute. The first thing an operating system must do is initialize all of its resources. For instance, it must locate the disk and set up the computer's memory so it can operate. Once the operating system has initialized the system, it is ready to accept input from the users.

A-2. SLIDE: What Happens on Bootup?

Purpose

To describe what the computer does when you turn the power on.

Key Points

Boot ROM tests the hardware. The secondary loader loads the operating software (OS). The OS initializes resources so users can access the system.

Teaching Question

What does "system startup" mean and what happens when a system is started up?

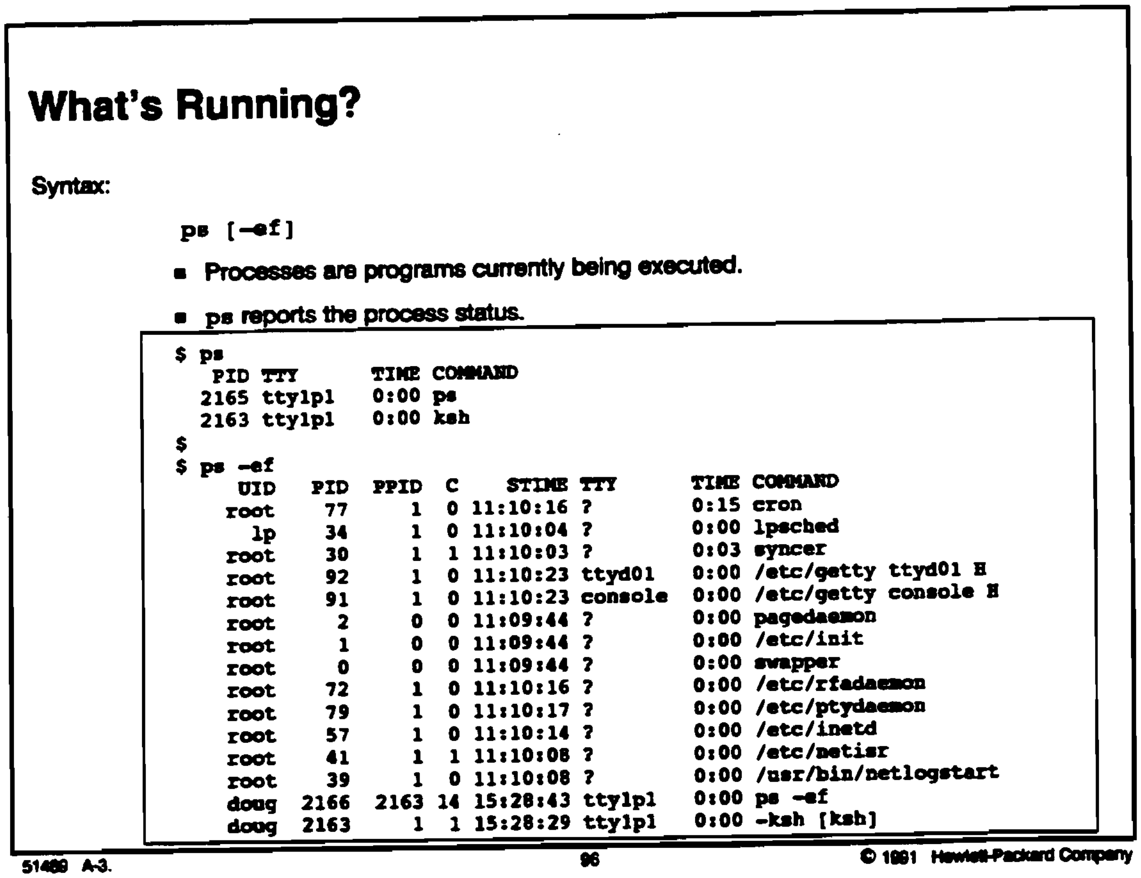
Presentation Suggestions

Students have probably seen a system booting before; however, if you could reboot your system it would help them a great deal to see different phases of system startup.

Transition

When the system is booted, there is a large number of things going on at once. The ps command can help us see what is happening on the system "under the hood."

A-3. SLIDE: What's Running?



What's Running?

Syntax:

```
ps [-ef]
```

- Processes are programs currently being executed.
- ps reports the process status.

```
$ ps
   PID TTY      TIME COMMAND
  2165 ttylpl   0:00 ps
  2163 ttylpl   0:00 ksh
$
$ ps -ef
     UID   PID  PPID  C    STIME TTY       TIME COMMAND
    root    77     1  0 11:10:16 ?         0:15 cron
      lp    34     1  0 11:10:04 ?         0:00 lpsched
    root    30     1  1 11:10:03 ?         0:03 syncer
    root    92     1  0 11:10:23 ttyd01    0:00 /etc/getty ttyd01 H
    root    91     1  0 11:10:23 console   0:00 /etc/getty console H
    root     2     0  0 11:09:44 ?         0:00 pagedaemon
    root     1     0  0 11:09:44 ?         0:00 /etc/init
    root     0     0  0 11:09:44 ?         0:00 swapper
    root    72     1  0 11:10:16 ?         0:00 /etc/rfadaemon
    root    79     1  0 11:10:17 ?         0:00 /etc/ptydaemon
    root    57     1  0 11:10:14 ?         0:00 /etc/inetd
    root    41     1  1 11:10:08 ?         0:00 /etc/netisr
    root    39     1  0 11:10:08 ?         0:00 /usr/bin/netlogstart
    doug  2166  2163 14 15:28:43 ttylpl    0:00 ps -ef
    doug  2163     1  1 15:28:29 ttylpl    0:00 -ksh [ksh]
```

51489 A-3. 96 © 1991 Hewlett-Packard Company

Student Notes

There is always something happening on your computer. Even if you are the only user and you are doing nothing, the system is constantly performing activities to keep the system running. The activities on a system are called **processes**. A process is a program that is doing work on the system.

As a user, you should keep track of everything that you are running on the system. The ps command (ps means process status) lists the active processes on a system.

Just typing ps will list the processes *you* are running on your *current* terminal. To see *everything* that is running, type ps -ef. As you may have guessed, the e option means list everything, and the f option gives you a full listing.

Notice that with the full listing you can see the parent of a particular process. On the slide, the ps commands are being run from the K-shell (ksh).

A-3. SLIDE: What's Running?

Instructor Notes

Purpose

To describe how a user can see all of the programs that are running on the system.

Key Points

Every process has a unique process id (PID). You can see this PID using any of the ps options.

Teaching Question

Why would you want to see what you are running on the system? Answer: To keep track of what processes you are currently running.

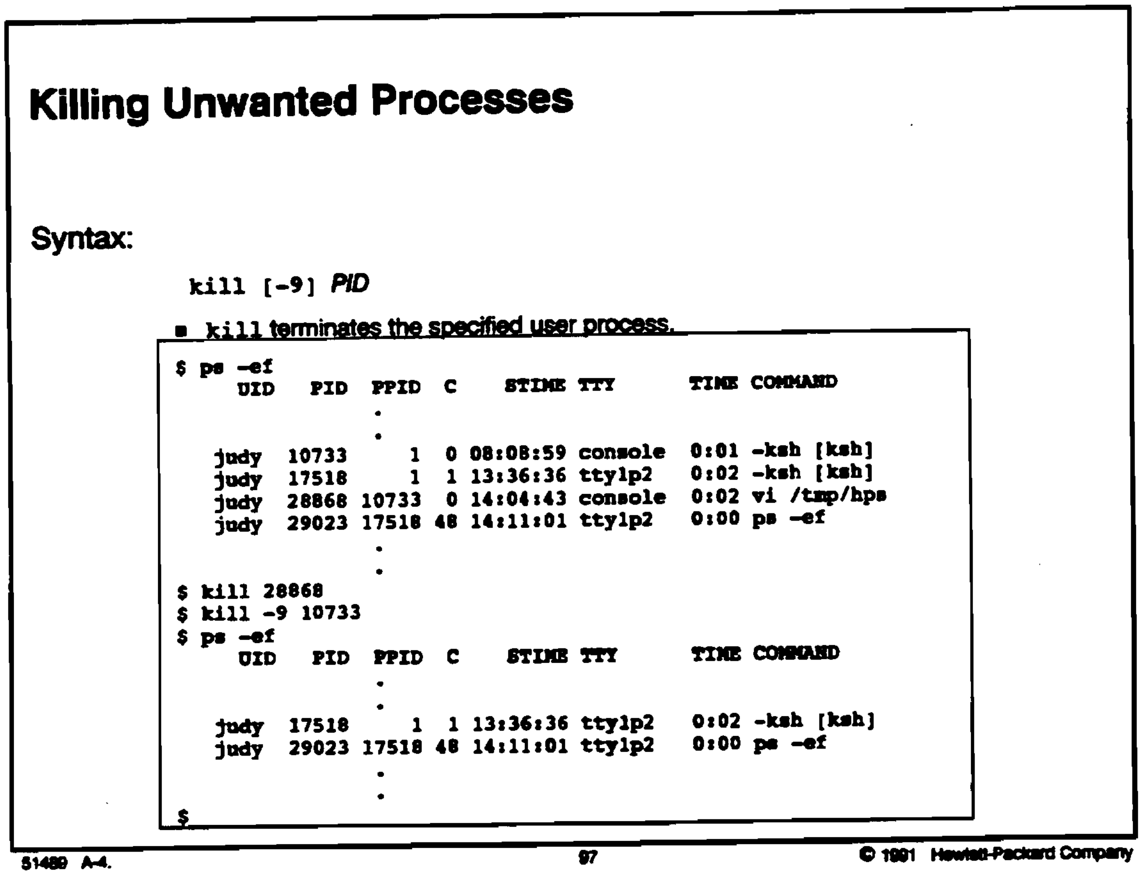
Evaluation Question

Where is the PID in the ps listing? Answer: It is in the first column.

Transition

How can you stop a process that is running?

A-4. SLIDE: Killing Unwanted Processes



Killing Unwanted Processes

Syntax:

```
kill [-9] PID
```

- `kill` terminates the specified user process.

```
$ ps -ef
     UID    PID  PPID  C    STIME TTY        TIME COMMAND
                       .
                       .
   judy  10733     1  0 08:08:59 console  0:01 -ksh [ksh]
   judy  17518     1  1 13:36:36 ttylp2   0:02 -ksh [ksh]
   judy  28868 10733  0 14:04:43 console  0:02 vi /tmp/hps
   judy  29023 17518 48 14:11:01 ttylp2   0:00 ps -ef
                       .
                       .
$ kill 28868
$ kill -9 10733
$ ps -ef
     UID    PID  PPID  C    STIME TTY        TIME COMMAND
                       .
                       .
   judy  17518     1  1 13:36:36 ttylp2   0:02 -ksh [ksh]
   judy  29023 17518 48 14:11:01 ttylp2   0:00 ps -ef
                       .
                       .
$
```

51489 A-4. 97 © 1991 Hewlett-Packard Company

Student Notes

Every process that ps reports has a PID. If you need to stop a process or if a program gets "hung" so that it no longer works, you can stop the process using the `kill` command:

```
$ kill PID
```

where PID is the process id reported by ps. If you run the ps command and the process you killed is still there, you can try `kill -9` *PID*. This is called a **sure kill**. If the process still exists in the ps listing, it is locked into the system and the only way to get rid of it is to reboot the system.

Note that you can only kill your own processes. Only the system administrator can kill a process belonging to someone else.

Caution

Some processes should never be killed with `kill -9`. They may die without freeing system resources. In this case, the only way to get the system back to normal is to reboot. Always try `kill` *PID* first.

A-4. SLIDE: Killing Unwanted Processes

Instructor Notes

Purpose

There are times when you want to stop a process. The kill command is the easiest way to do that.

Key Points

Always try to `kill` a process first. A process such as `/etc/inetd` will not free system resources if it is killed with the `-9` option.

A user can only kill his or her own process.

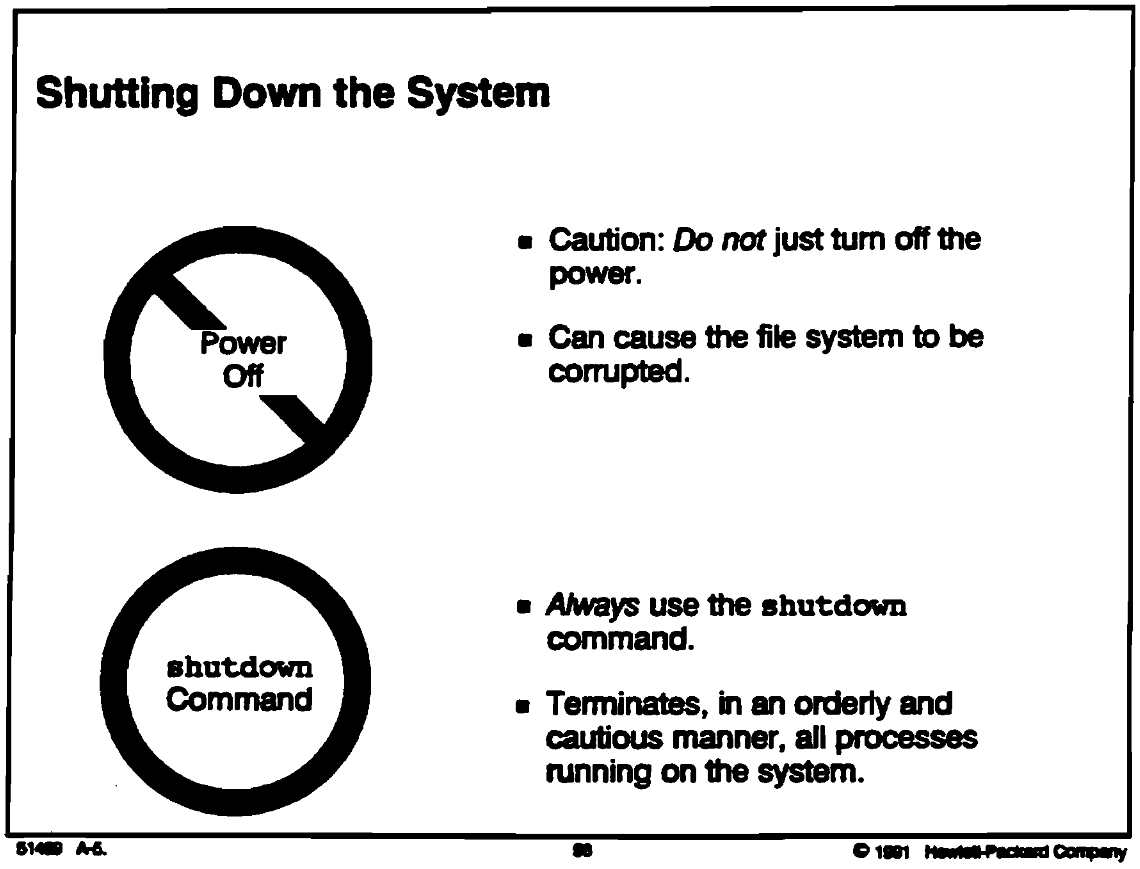
Where Problems Arise

If the user is on a workstation using only one display and their process "hangs" the terminal, then there is no way to kill it to regain control. If the machine is in a network environment, it may be possible to remotely kill the process. However, this should be left to the system administrator.

Transition

If we want to kill all running processes, we have to log in as the superuser (called "root") and shut down the system.

A-5. SLIDE: Shutting Down the System



Shutting Down the System

Power
Off

- Caution: *Do not* just turn off the power.
- Can cause the file system to be corrupted.

shutdown
Command

- *Always* use the `shutdown` command.
- Terminates, in an orderly and cautious manner, all processes running on the system.

51469 A-5. 98 © 1991 Hewlett-Packard Company

Student Notes

Most of the time, the HP-UX system is in a mode that allows many user and system processes to run simultaneously. Because there are so many processes running on the system at any given time, you should *never* just shut off the power to the computer. If a UNIX system is shut down improperly, you run the risk of damaging the data on your disk.

There are several reasons for shutting down your system. If a process is "hung" or acting strangely, you can try to `kill` it. If this does not work, however, you may have to shut down the system and reboot. You may also want to turn off the system if you are leaving it unattended for an extended period of time (such as over a weekend or vacation). As we have said, you should never turn off a UNIX system without shutting it down properly first.

There are several different ways to properly shut down an HP-UX system. Care should be taken so that any shutdown procedure results in a consistent, orderly stopping of system activities. Pulling the plug on the machine is obviously not a proper shutdown method. The `shutdown` command is normally used to shut down an HP-UX system.

Note

You must be logged in as *root* (the superuser) or be included in the authorization file, `shutdown.allow`, and have a current working directory of "/" in order to properly shut down the system. If you are not root and your name is not included in `shutdown.allow`, you should not power off the system unless it is an emergency!

A-5. SLIDE: Shutting Down the System

Purpose

To describe the reasons for properly shutting down an HP-UX system.

Key Points

If processes are active and accessing the disk when the system is turned off, it is more than likely that the data on the disk will become corrupt.

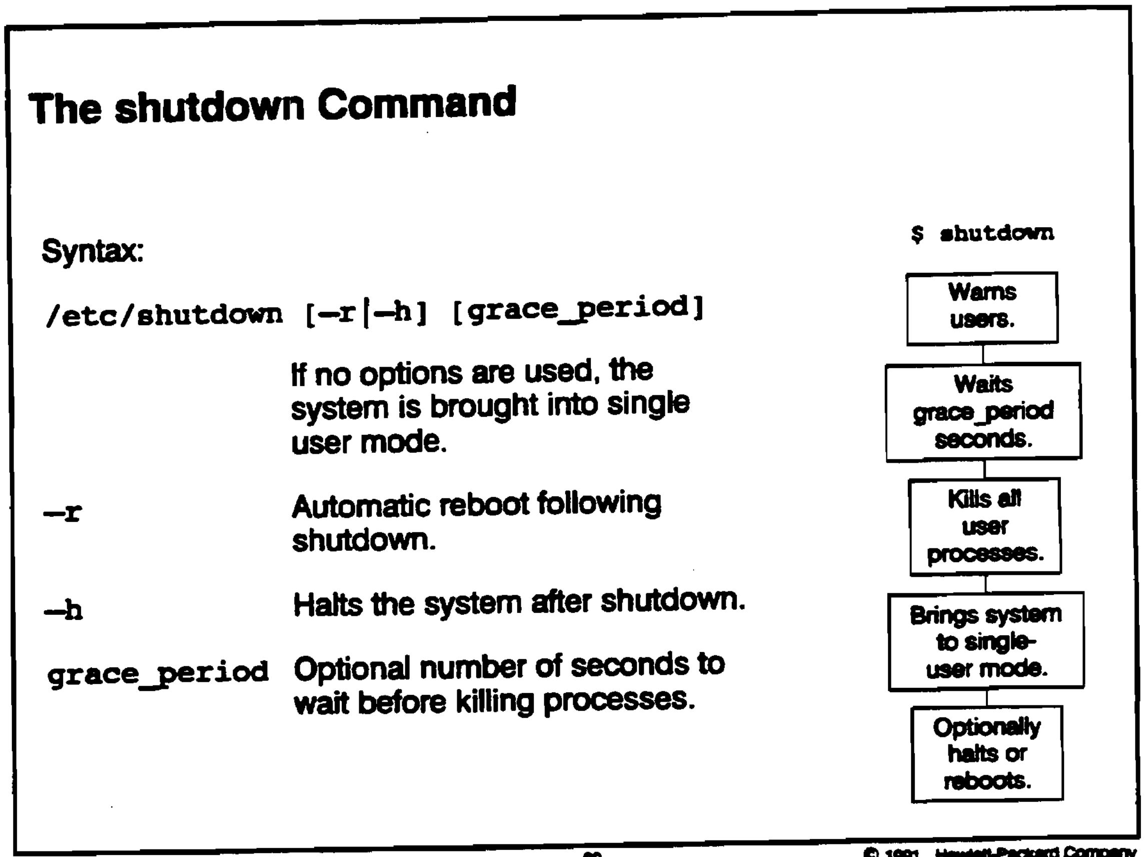
Evaluation Question

Why would you want to shut down your system?

Transition

`shutdown` is the command we will use to properly shut down the system.

A-6. SLIDE: The shutdown Command

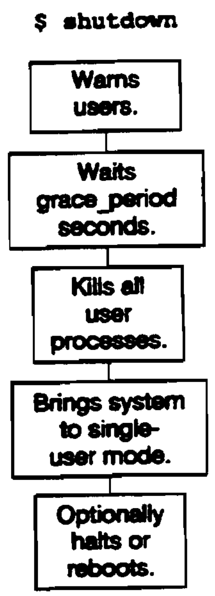


The shutdown Command

Syntax:

`/etc/shutdown [-r|-h] [grace_period]`

	If no options are used, the system is brought into single user mode.
`-r`	Automatic reboot following shutdown.
`-h`	Halts the system after shutdown.
`grace_period`	Optional number of seconds to wait before killing processes.


$ shutdown
Warns users.
Waits grace_period seconds.
Kills all user processes.
Brings system to single-user mode.
Optionally halts or reboots.

51489 A-6. 90 © 1991 Hewlett-Packard Company

Student Notes

The **shutdown** command results in an orderly and consistent cessation of all user activities. All active user processes are stopped so administrative activities can occur on a "quiet" system. This "quiet" state is called **single-user mode** because only the superuser can be logged in (usually at the system console). Remember, you must be *root* or be included in `shutdown.allow` in order to use the shutdown command.

The format of the command looks like this:

```
/etc/shutdown [ -r | -h ] [ grace ]
```

If shutdown is invoked with no options, the system is quietly brought down to a "quiet" (single-user) state, and you can do whatever activities may be necessary at that time. The options are explained as follows:

`-r`	Reboot the system automatically after reaching single-user mode. This is a good way to reset the system if something goes wrong.

`-h`	Halt the system after reaching the single-user mode. This is the *only* time you may turn off the power to the computer.
grace	This is the number of seconds to wait before actually shutting down the system. This gives other users a chance to save their work and log off. If you do not specify *grace*, the system will wait 60 seconds before shutting down.

When it is executed, the `shutdown` command does the following:

- It broadcasts a message to users that the system is being shut down in *grace* seconds. The individual invoking `shutdown` has the option of sending his or her own message to other users on the system.
- It waits the specified number of seconds.
- It stops all system programs such as accounting, error reporting, and line printer spooling.
- It terminates all other user processes running on the system using the `killall` command (refer to killall(1m)). This includes killing everyone's shell.
- It brings the system to single-user mode.
- It disconnects any disks other than the system disk.
- It optionally reboots or halts the system.

But what if you are not authorized to shut your system down (you are not included in `shutdown.allow`) and your system administrator is on vacation and did not leave the root password with someone? And what if you are told to turn off all of your computers because the power will be going out in the building shortly. Don't panic! If you cannot log in as root and are not included in `shutdown.allow`, and you *absolutely* have to turn the system off, follow these steps:

1. Tell everyone on the system to save their work and log off.
2. Make sure the printer has stopped printing.
3. Check that no one else is on the system by using the `ps` command.
4. Execute the `sync` command twice.
5. Turn off the power.

Note This is *not* a proper shutdown. However, it is the best you can do if you cannot log in as root and are not included in `shutdown.allow`. By making sure everyone is off the system (except you, of course) and executing the `sync` command, the system should recover when you reboot it.

A-6. SLIDE: The shutdown Command

Instructor Notes

Purpose

To introduce the proper way to shut down the system.

Key Points

Stress that the only time that you should cycle power on the CPU or disk drive is after the system has been halted. The last paragraph is included only if there is an emergency and the systems have to be turned off and no one can log in as root or is included in `shutdown.allow`.

Evaluation Question

Ask the class why they would want to use shutting down the system to reboot HP-UX. Answer: It is a good way to reset the system if something has gone wrong.

Why do you have to halt the system before turning it off? Answer: All activities on the system must be stopped in an orderly, consistent manner.

Transition

Here are some exercises to practice using `ps` and to help you understand the shutdown procedure.

A-7. LAB: Exercises

1. Execute the ps command to show all processes running on the system. How many processes are running? (Use a pipe!) How many processes are you running?

2. Looking at the listing, can you tell what PPID means?

3. If you are running `vi` and your terminal locks up, how can you fix it?

4. Who are the only people who can properly shut down the system?

5. What does the **-h** option to the **shutdown** command do?

6. Why is it important that the system be quiet when you shut it down?

7. Write (*don't execute*) the command to shut down the system to a halt in 30 seconds.

A-7. LAB: Exercises

Purpose

To practice finding processes and to reinforce the concept of a proper shutdown.

Key Points

Since students cannot be the superuser unless you let them, these exercises are designed so that they can get the most out of what they can do. If each student (or student pair) has their own workstation, you may want to include the class in `shutdown.allow`. Don't forget to include *root* in `shutdown.allow`. Also, make sure that `/etc/shutdown` has execute permission.

Transition

You may choose another optional module.

Solutions

1. Execute the ps command to show all processes running on the system. How many processes are running? (Use a pipe!) How many processes are you running?

Answer:

```
$ ps -ef
     UID   PID  PPID  C    STIME TTY       TIME COMMAND
    root    38     1  0   Jul 25 ?         0:13 /etc/cron
    root    97     1  0   Jul 25 console   0:00 /etc/getty console console
    root     4     0  0   Jul 25 ?         2:11 netisr
    root     3     0  0   Jul 25 ?         0:02 statdaemon
    root     2     0  0   Jul 25 ?         0:00 pagedaemon
    root     1     0  0   Jul 25 ?         0:07 init
    root     0     0  0   Jul 25 ?         0:04 swapper
    root   103     1  0   Jul 25 tty1p0    0:00 /etc/getty -h tty1p0 9600
    root  1215     1  0   Jul 26 tty1p1    0:00 /etc/getty -h tty1p1 9600
    bill   116     1  0   Jul 25 tty3p2    0:06 -ksh [ksh]
    bill  1932   116  1 10:54:36 tty3p2    0:07 vi mod11
    bill  1994  1932 70 11:39:24 tty3p2    0:03 ps -ef
  .
  .
         etc
```

You can see every process that is running. Simply count them to find out how many there are. The pipe to count processes could be `ps -ef | wc -l`. This would also count the header line.

Look through the ps listing to find processes with your name in the UID column.

2. Looking at the listing, can you tell what PPID means?

Answer:

PPID is parent process id. This is the process that started the process you are looking at.

3. If you are running `vi` and your terminal locks up, how can you fix it?

Answer:

Log in to another terminal, execute ps to find the PID of the hung `vi` process, and execute the **`kill`** PID command to kill it.

4. Who are the only people who can properly shut down the system?

Answer:

The root (the superuser) and/or anyone who is given authorization by being included in the `shutdown.allow` file.

5. What does the `-h` option to the `shutdown` command do?

Answer:

It halts the system.

6. Why is it important that the system be quiet when you shut it down?

Answer:

The more activity there is on a system, the greater chance you have of losing your work when the system goes down.

7. Write (*don't execute*) the command to shut down the system to a halt in 30 seconds.

Answer:

```
# shutdown -h 30
#
```

Notice that the prompt is a "#". This is the prompt the superuser gets when he or she logs in.

Appendix B — Using the System with Key Shell

Objectives

Upon completion of this module, you will be able to do the following:

- Use Key Shell softkeys to perform HP-UX commands.
- Translate a softkey command into HP-UX syntax.
- Use command line editing to correct mistakes.
- Use online help for pre-configured softkeys and selected topics.

Appendix B — Using the System with Key Shell

Overview of Module B

This module will introduce students to the Key Shell. It describes how to use the Key Shell to enter basic HP-UX commands through a menu of softkeys that resemble regular English.

Motivation

Because the Key Shell is a menu-based softkey interface to the Korn Shell, it allows users with little HP-UX experience to execute basic commands and operate on files with relative ease. When these softkey commands are executed, they are translated into standard HP-UX; this helps the user to learn HP-UX syntax.

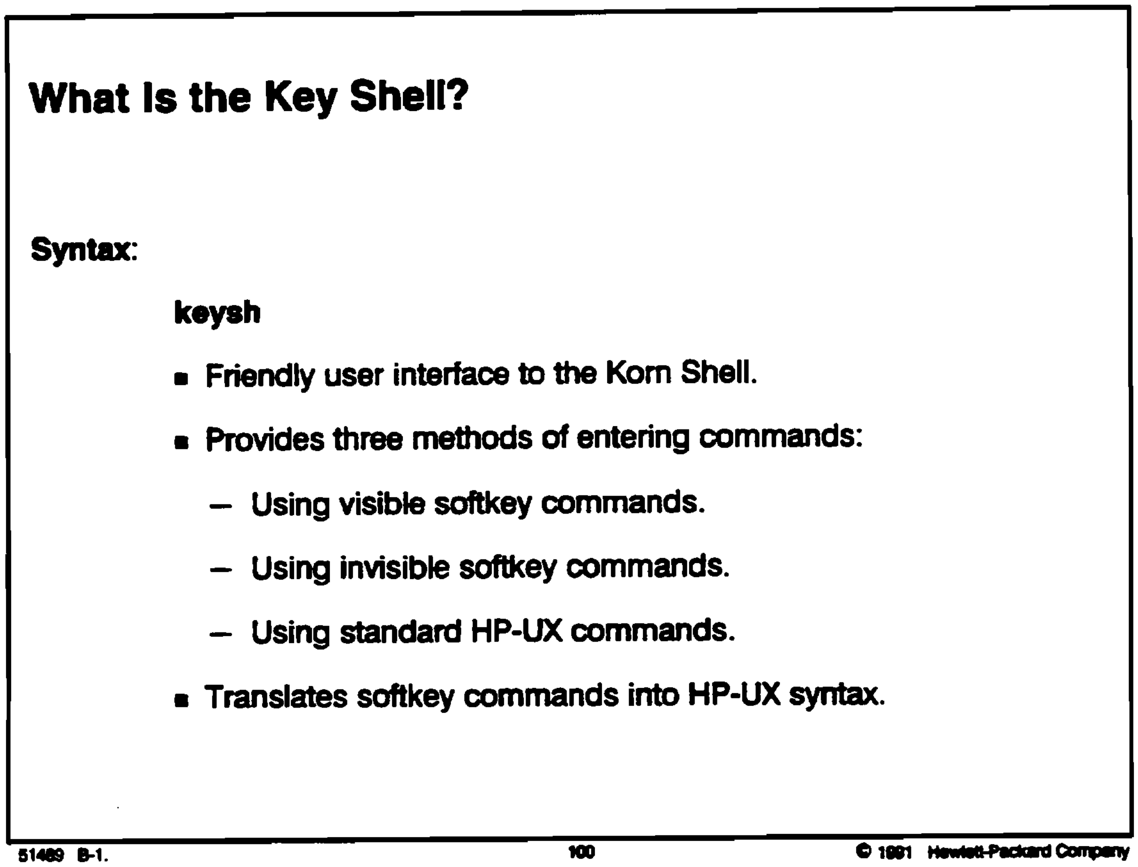
Exercises

There are short exercises located throughout this module. There are also exercises at the end of the module.

Module Transition

What optional module you discuss next is your choice.

B-1. SLIDE: What Is the Key Shell?



What Is the Key Shell?

Syntax:

keysh

- Friendly user interface to the Korn Shell.
- Provides three methods of entering commands:
  - Using visible softkey commands.
  - Using invisible softkey commands.
  - Using standard HP-UX commands.
- Translates softkey commands into HP-UX syntax.

51489 B-1. 100 © 1991 Hewlett-Packard Company

Student Notes

The Key Shell was developed by Hewlett-Packard to provide an easy-to-use interface to the Korn Shell by allowing users to build softkey commands for basic HP-UX commands. It provides softkey menus and online help to aid you in manipulating files and building basic HP-UX commands.

To start the Key Shell, you simply type `keysh` at the shell prompt. To leave the Key Shell, you type `exit` or Ctrl-d.

B-1. SLIDE: What Is the Key Shell?

Instructor Notes

Purpose

Some applications users will like using the Key Shell as their interface to the system. For novice HP-UX users, it eliminates the fear of having to know the many HP-UX commands.

Key Points

The Key Shell is a friendly way to use the system for running applications, manipulating files, and executing basic HP-UX commands. You may also point out that this is a Hewlett-Packard value-added capability available on HP systems beginning with HP-UX 8.0.

Where Problems Arise

In order for the Key Shell screen to function properly, `$TERM` must be set to the terminal type you are using. One way to do this is to add the following line to `.profile`:

```
TERM='tset - -Q'
```

If your terminal is not a standard size, the `$LINES` and `$COLUMNS` must be set to the correct value for your terminal.

Teaching Question

Ask how you would invoke the Key Shell if you were not in it.

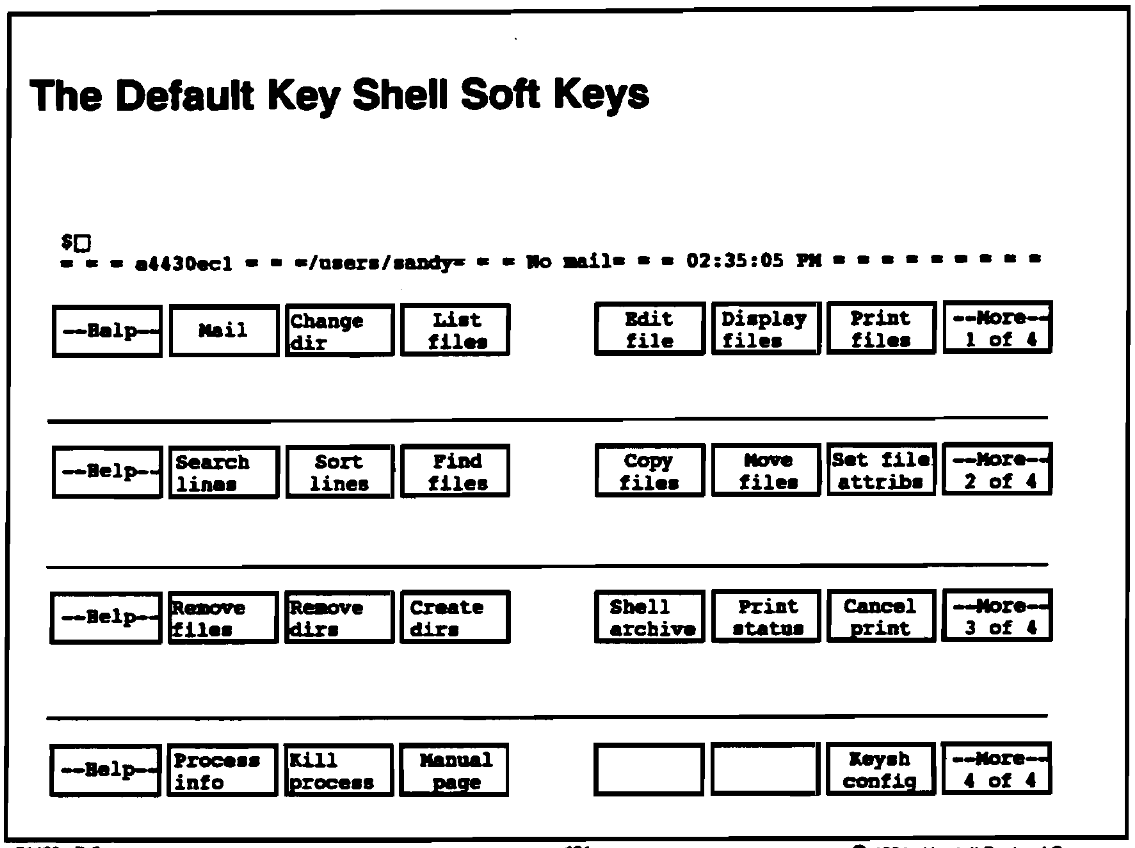
Presentation Suggestions

Have the students run the Key Shell and ask them what they see.

Transition

Let's look at the default Key Shell softkeys.

B-2. SLIDE: The Default Key Shell SoftKeys



The Default Key Shell Soft Keys

```
$□
= = = a4430ec1 = = =/users/sandy= = = No mail= = = 02:35:05 PM = = = = = = = = = =

|--Help--| Mail  |Change |List  |   |Edit  |Display|Print  |--More--|
|        |       |dir    |files |   |file  |files  |files  |1 of 4  |

|--Help--|Search |Sort   |Find  |   |Copy  |Move   |Set file|--More--|
|        |lines  |lines  |files |   |files |files  |attribs |2 of 4  |

|--Help--|Remove |Remove |Create|   |Shell |Print  |Cancel |--More--|
|        |files  |dirs   |dirs  |   |archive|status|print  |3 of 4  |

|--Help--|Process|Kill   |Manual|   |      |       |Keysh  |--More--|
|        |info   |process|page  |   |      |       |config |4 of 4  |
```

51489 B-2 101 © 1991 Hewlett-Packard Company

Student Notes

When you first execute **keysh**, you will see a display like that in the top figure. The first line is your standard Korn Shell prompt, **$**. Underneath that is the status line; by default this displays the host name, the current directory, the mail status, and the time. The next line shows the top bank of *visible* softkeys. Each softkey corresponds to the function keys, **f1** through **f8** on your keyboard. If you are in a window environment, you will see **hpterm** in the center separating the keys into groups of four; otherwise this will be blank. There are four banks of available softkeys. By selecting **--More--** several times, you can access the other *visible* softkeys.

The following table describes the *visible* softkey commands that are configured in **keysh**:

Softkey	Function
—Help—	Gives online help for all pre-configured softkeys and their options.
Mail	Processes electronic mail interactively.
Change dir	Changes the current directory.
List files	Lists the contents of a directory.
Edit files	Edits files on a screen-oriented display.
Display files	Displays the contents of a file one screen at a time.
Print files	Formats a file and sends it to the line printer.
Search lines	Searches for lines matching a pattern.
Sort lines	Sorts the lines of a file.
Find files	Locates files within a directory.
Copy files	Copies files to another location.
Move files	Moves or renames a file.
Set file attribs	Changes permissions, owner, or group of a file.
Remove files	Deletes a file.
Remove dirs	Deletes a directory.
Create dirs	Creates a new directory.
Shell archive	Bundles one or more files into a shell archive package for mailing or moving.
Print status	Shows current status of all printers.
Cancel print	Cancels a print request.
Process info	Shows status of active processes.
Kill process	Terminates a process.
Manual page	Accesses the online manual pages.
Keysh config	Configures the appearance and behavior of **keysh**.

B-2. SLIDE: The Default Key Shell SoftKeys

Instructor Notes

Purpose

Before you begin using keysh, you need to know what functions the default *visible* softkeys provide.

Key Points

The purpose of a softkey is to make your job easier. Instead of typing a command, you use a softkey to perform the function you wish to execute.

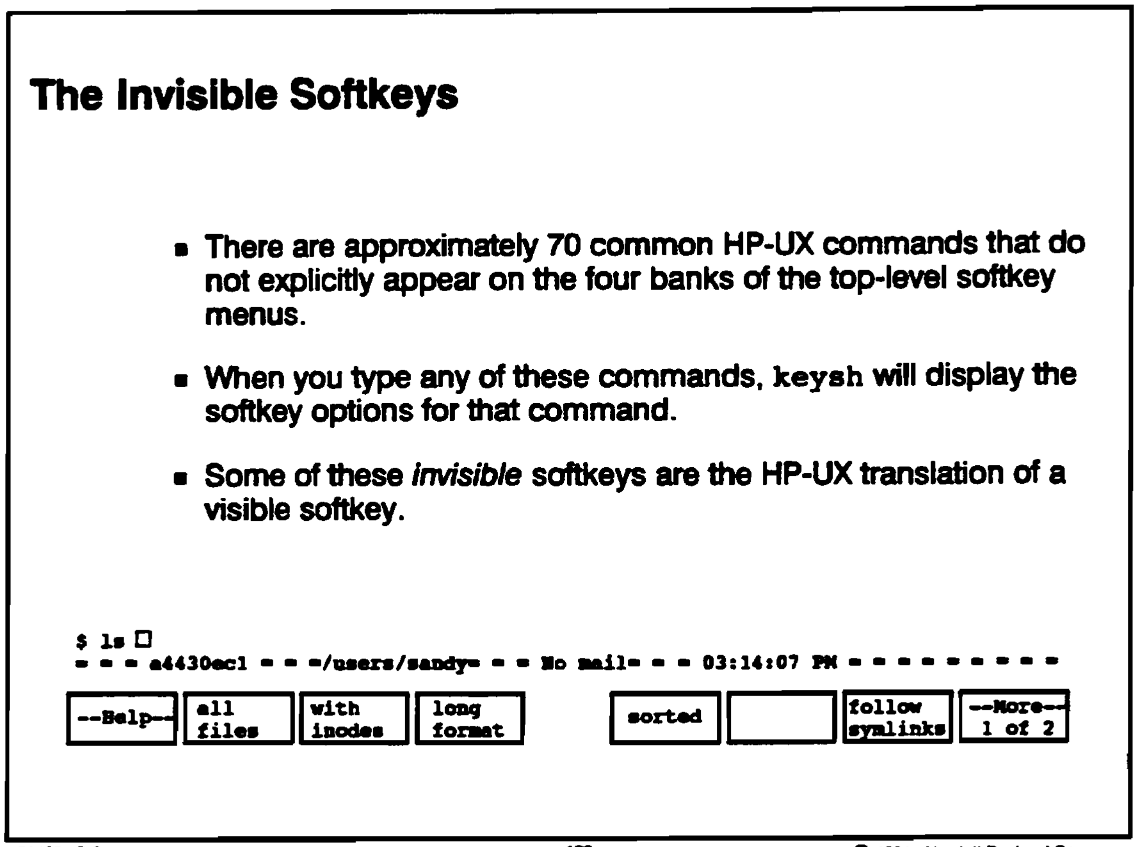
Presentation Suggestions

Have students cycle through the four banks of softkeys so they can see what is available.

Transition

Additional commands are available using the *invisible* softkeys.

B-3. SLIDE: The Invisible Softkeys



The Invisible Softkeys

- There are approximately 70 common HP-UX commands that do not explicitly appear on the four banks of the top-level softkey menus.
- When you type any of these commands, **keysh** will display the softkey options for that command.
- Some of these *invisible* softkeys are the HP-UX translation of a visible softkey.

```
$ ls □
= = = a4430ecl = = =/users/sandy= = = No mail= = = 03:14:07 PM = = = = = = = = =
```

51489 B-3. 102 © 1991 Hewlett-Packard Company

Student Notes

When you type a recognized HP-UX command, **keysh** will automatically change the default softkeys to the available softkey options for the command entered. There may be more than one bank of softkeys for these options.

The following is a list of commands that we have covered in this course that are available as invisible softkey commands:

cancel	ls
cat	man
cd	mkdir
chgrp	more
chmod	pr
chown	ps
exit	rm
find	rmdir
kill	vi
lp	wc
lpstat	who

B-3. SLIDE: The Invisible Softkeys

Instructor Notes

Purpose

Before you begin using keysh, you need to know that keysh recognizes many HP-UX commands that do not appear on the top-level softkey menus.

Key Points

When these *invisible* softkey commands are typed from the keyboard, keysh automatically displays the appropriate softkey options. Some of these commands are HP-UX translations of the *visible* softkeys. This means that when a *visible* softkey is executed, keysh will display the same softkey options that would be displayed after entering an *invisible* softkey.

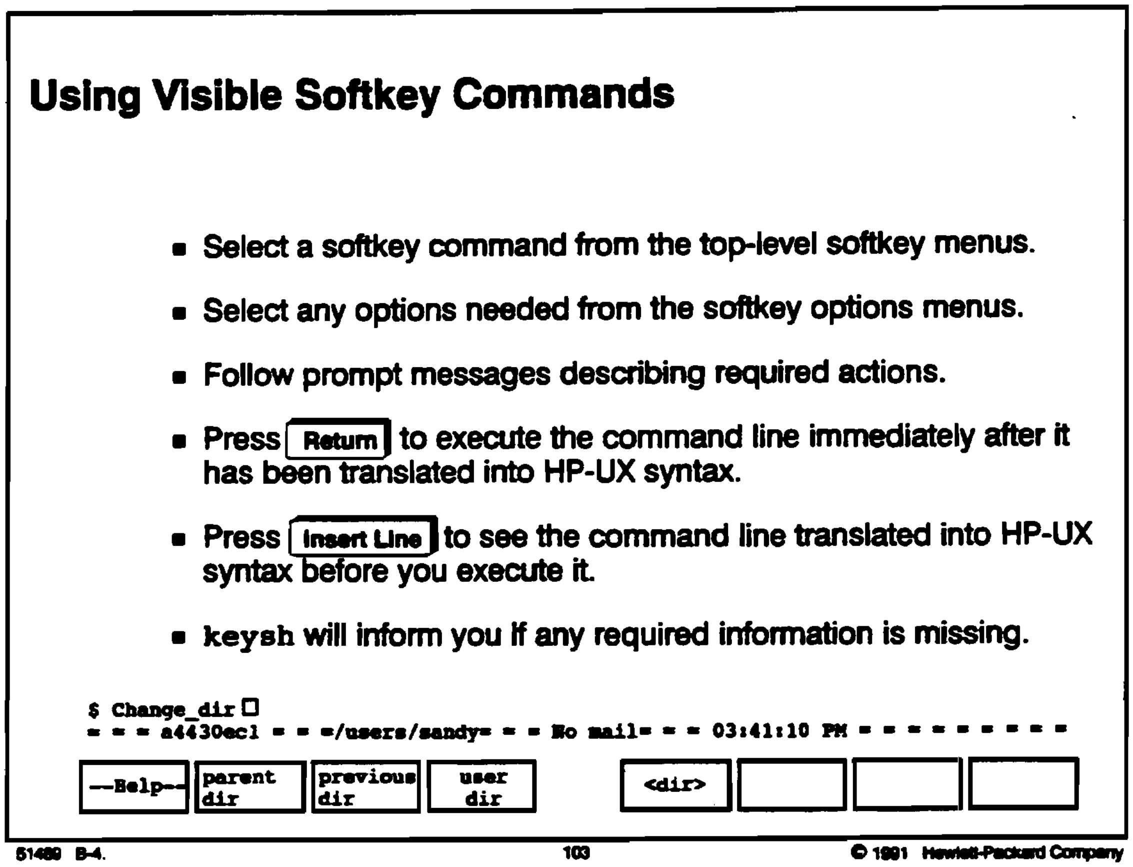
Presentation Suggestions

Have the students type in an *invisible* command so that they can see the softkey options that keysh invokes.

Transition

Let's see how to use a *visible* softkey command.

B-4. SLIDE: Using Visible Softkey Commands



Using Visible Softkey Commands

- Select a softkey command from the top-level softkey menus.
- Select any options needed from the softkey options menus.
- Follow prompt messages describing required actions.
- Press Return to execute the command line immediately after it has been translated into HP-UX syntax.
- Press Insert Line to see the command line translated into HP-UX syntax before you execute it.
- **keysh** will inform you if any required information is missing.

```
$ Change_dir □
= = = a4430ecl = = =/users/sandy= = = No mail= = = 03:41:10 PM = = = = = = = = = =
```

51489 B-4. 103 © 1991 Hewlett-Packard Company

Student Notes

Select a softkey by pressing the corresponding function key or clicking on the softkey with your mouse. Always select softkeys from left to right; do not attempt to insert words or options out of order in the command line. Using the `--More--` softkey allows you to toggle through the banks of softkeys. Once you have chosen a *visible* softkey, the option softkeys appear. Choosing an option softkey inserts the corresponding command or option into the command line. **String softkeys** are enclosed in angle brackets and indicate that you need to type text, such as a *user_name* or *file_name* on the command line.

To see a softkey command translated into HP-UX syntax before being executed, use the Insert Line key. To execute the command directly, hit Return; the translated command will appear highlighted on the screen and will be executed.

Exercises

For each of the following exercises, use the *visible* softkeys and the option softkeys to enter the command.

1. Get a long listing of the contents of your login directory. Look at the translation of the softkeys before executing the command.

2. Change the permission of the file `mutant` so you have read and write permission and everyone else just has read permission. Execute the command immediately.

B-4. SLIDE: Using Visible Softkey Commands — Instructor Notes

Purpose

For those users unfamiliar with HP-UX syntax, using the *visible* softkeys and the option softkeys allows them to construct HP-UX commands and manipulate files.

Key Points

Point out that pressing the (Insert Line) key translates the softkey command into HP-UX syntax; it does not execute the command. If you wish to execute the command, you must then press (Return). If you press (Return) after completing the softkey command, the translation of the command will appear highlighted on the screen and will be executed.

Transition

Let's see how to use an *invisible* softkey command.

Answers

For each of the following exercises, use the *visible* softkeys and the option softkeys to enter the command.

1. Get a long listing of the contents of your login directory. Look at the translation of the softkeys before executing the command.

Answer:

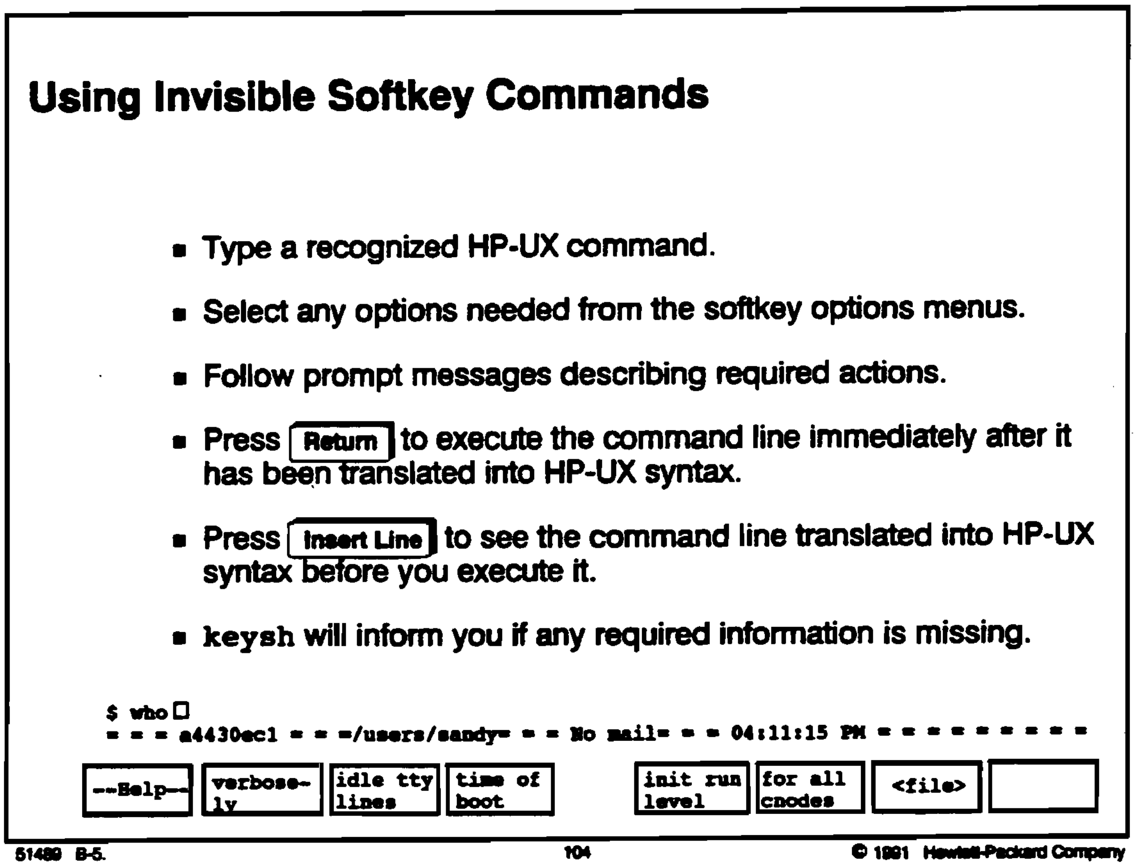
Press `List files` from the first bank of softkeys. Press `long format` from the option softkey menu. Press (Insert Line) to see the translation and then press (Return).

2. Change the permission of the file `mutant` so you have read and write permission and everyone else just has read permission. Execute the command immediately.

Answer:

Toggle the `--More--` key to get to the second bank of softkeys, and press `Set file attribs`. Press `mode` from the option softkey menu, then press `readable by`, `all`, `writable by`, and `owner` in this order. Notice that each time you press a key from the option softkey menu, a new option softkey menu appears. Finally, type the name of the file, `mutant`, and press (Return).

B-5. SLIDE: Using Invisible Softkey Commands



Using Invisible Softkey Commands

- Type a recognized HP-UX command.
- Select any options needed from the softkey options menus.
- Follow prompt messages describing required actions.
- Press Return to execute the command line immediately after it has been translated into HP-UX syntax.
- Press Insert Line to see the command line translated into HP-UX syntax before you execute it.
- **keysh** will inform you if any required information is missing.

```
$ who□
= = = a4430ec1 = = =/users/sandy= = = No mail= = = 04:11:15 PM = = = = = = = = = =
```

51489 B-5. 104 © 1991 Hewlett-Packard Company

Student Notes

Type in a recognized HP-UX command at the **$** prompt. Once you have typed in the final character of this command, the option softkeys will appear. Choosing an option softkey inserts the corresponding command or option into the command line. Do not mix HP-UX options and softkey options on the same command line. String softkeys are enclosed in angle brackets and indicate that you need to type text, such as a *user_name* or *file_name* on the command line.

To see a softkey command translated into HP-UX syntax before being executed, use the Insert Line key. To execute the command directly, hit Return; the translated command will appear highlighted on the screen and will be executed.

Exercises

For each of the following exercises, use the *invisible* softkey commands and the option softkeys.

1. Using the `who` *invisible* softkey, list those login lines that are not currently being used. Execute the command immediately.

2. Find out how many lines are in the file `new.jersey`. Look at the the translation of the softkeys before executing the command.

B-5. SLIDE: Using Invisible Softkey Commands Instructor Notes

Purpose

For those users with a little more HP-UX experience, the *invisible* softkeys give access to a greater number of HP-UX commands. The softkey options can also act as "memory joggers" for those commands that are not frequently used.

Key Points

Point out that the option softkeys will appear immediately after the last character of a recognized HP-UX command has been typed.

Transition

How do you correct mistakes in the Key Shell?

Answers

For each of the following exercises, use the *invisible* softkey commands and the option softkeys.

1. Using the `who` *invisible* softkey, list those login lines that are not currently being used. Execute the command immediately.

Answer:

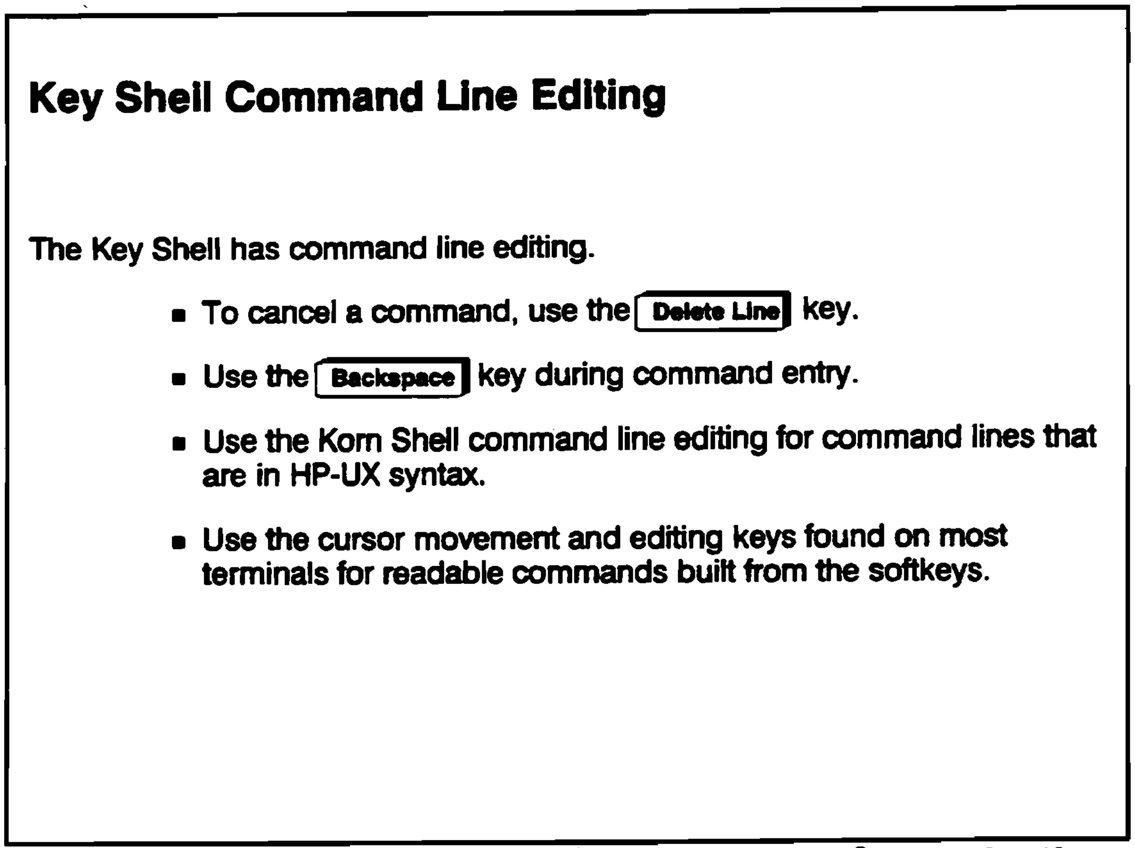
Type `who` at the $ prompt. When the softkey options appear, press `idle tty lines` ; then press (Return).

2. Find out how many lines are in the file `new.jersey`. Look at the the translation of the softkeys before executing the command.

Answer:

Type `wc` at the $ prompt. When the softkey options appear, press `lines` ; then type the name of the file, `new.jersey`. Finally, press (Insert Line) to see the translation and then press (Return)

B-6. SLIDE: Key Shell Command Line Editing



Key Shell Command Line Editing

The Key Shell has command line editing.

- To cancel a command, use the Delete Line key.
- Use the Backspace key during command entry.
- Use the Korn Shell command line editing for command lines that are in HP-UX syntax.
- Use the cursor movement and editing keys found on most terminals for readable commands built from the softkeys.

51489 B-6. 105 © 1991 Hewlett-Packard Company

Student Notes

Key Shell allows extensive command line editing. You can edit either of two types of command lines:

- The readable command line built by using the softkeys.
- The HP-UX command line that you typed, or that appeared after you pressed Insert Line.

You can edit the readable command line built by using the softkeys even after you have pressed Insert Line to translate it to HP-UX syntax. Just press ▲ to retrieve the softkey command from the command buffer.

You can edit a command line by using the command line editing available with the Korn Shell (`vi`) or the cursor movement and editing keys found on most terminals. Remember to press Esc before using `vi` editing. Use caution when mixing `vi` and key editing on the same command line.

B-6. SLIDE: Key Shell Command Line Editing

Instructor Notes

Purpose

Key Shell users need to be able to correct any mistakes made in the command line.

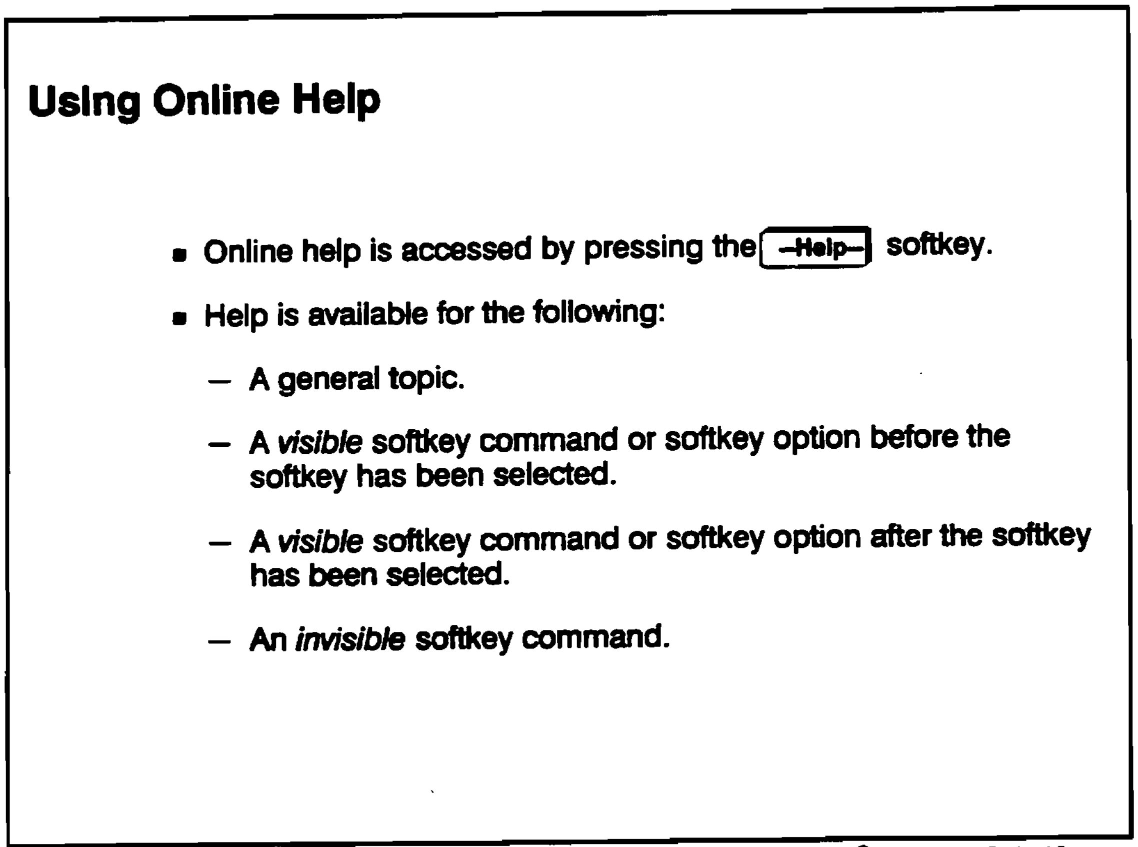
Key Points

Stress that it is important to press Esc before using `vi` editing. Also, unusual "happenings" may occur if both `vi` and the editing keys are used to correct a command line.

Transition

Help is available while using **keysh**

B-7. SLIDE: Using Online Help



Using Online Help

- Online help is accessed by pressing the -Help- softkey.
- Help is available for the following:
  - A general topic.
  - A *visible* softkey command or softkey option before the softkey has been selected.
  - A *visible* softkey command or softkey option after the softkey has been selected.
  - An *invisible* softkey command.

51489 B-7. 106 © 1991 Hewlett-Packard Company

Student Notes

Online help is available for all pre-configured softkeys. To get help for a *visible* softkey command or softkey option before the softkey has been selected, press `--Help--` , then press the desired softkey. To get help for a *visible* softkey command or softkey option after the softkey has been selected, press `--Help--` , then press Return. To get help for an *invisible* softkey command, type the command, press `--Help--` , then press Return.

In addition, the following help topics are available by pressing `--Help--` twice:

`using help`	How to use the online help.
`using keysh`	How to use the Key Shell.
`editing`	Editing the command line.
`visibles`	Visible softkeys.
`invisibles`	Invisible softkeys.
`keysh errors`	Key Shell error messages.
`regexp patterns`	Regular expressions and pattern matching.
`redirect pipe`	Command input/output redirection and piping.

B-7. SLIDE: Using Online Help

Instructor Notes

Purpose

This slide explains how to get help while in the Key Shell.

Key Points

Having help available online aids the student in constructing appropriate HP-UX commands. By giving an explanation for each softkey, constructing commands becomes relatively easy.

Transition

Let's do some exercises using `keysh`.

B-8. LAB: Exercises

Complete each of the following exercises using the Key Shell soft keys.

1. Notice the two files named `display1` and `display2` in your home directory. Print their contents to your terminal.

2. Make a copy of `display1` and call it "More_Than_14_chars". What happened?

3. Rename `display1` to `new.display1`.

4. Construct an HP-UX command that will copy `display2` to `new.display1` but will prompt you to verify that the file should be copied when copying it would overwrite an existing file.

5. Construct an HP-UX command that will find all files in your home directory that are larger than 1200 bytes.

6. Construct an HP-UX command that will display all the processes running on the system that belong to another member of the class. (You will need to know how that person logged in.)

B-8. LAB: Exercises

Instructor Notes

Purpose

Practice using `keysh` to construct HP-UX commands.

Transition

You may choose another optional module.

Solutions

Complete each of the following exercises using the Key Shell soft keys.

1. Notice the two files named `display1` and `display2` in your home directory. Print their contents to your terminal.

Answer:

Press `Display files` and then type `display1` and `display2`.

2. Make a copy of `display1` and call it "More_Than_14_chars". What happened?

Answer:

Press `Copy files` from the second bank of softkeys. Then type `display1`, press `to`, and type `More_Than_14_chars`. Return to the first bank of softkeys and press `List files`. You will notice that the file was truncated to 14 characters unless you are using long file names. If you are on system using long file names, the complete file name will be used.

3. Rename `display1` to `new.display1`.

Answer:

Press `Move files` from the second bank of softkeys. Then type `display1`, press `to`, and type `new.display1`.

4. Construct an HP-UX command that will copy `display2` to `new.display1` but will prompt you to verify that the file should be copied when copying it would overwrite an existing file.

Answer:

Press `Copy files` from the second bank of softkeys, then press `interactively` from the option softkeys that appear. Type `display2`, press `to`, and type `new.display1`. You should see the prompt `overwrite display2? (y/n)`.

5. Construct an HP-UX command that will find all files in your home directory that are larger than 1200 bytes.

Answer:

Press `Find files` from the second bank of softkeys, type the name of your home directory, press `size` from the second bank of option softkeys, press `greater than` , and type `1200`.

6. Construct an HP-UX command that will display all the processes running on the system that belong to another member of the class. (You will need to know how that person logged in.)

Answer:

Press `Process info` from the fourth bank of softkeys, press `for user` from the option softkeys that appear, and then type the name of the person for whom you are displaying running processes.

Appendix C — HP Visual User Environment (VUE)

Objectives

Upon completion of this module, you will be able to do the following:

- Explain what HP VUE is.
- Use the mouse to control the HP VUE environment.
- Use the HP VUE Workspace Manager.
- Work with terminal windows.
- Use the File Manager to manipulate files and directories.
- Use the Help Manager to obtain information on a topic.
- Use the Style Manager to customize the HP VUE environment.

Appendix C — HP Visual User Environment (VUE)

Overview of Module C

This module will introduce students to the HP Visual User Environment (VUE), the default windowing environment, that is supplied with HP-UX 8.0 on the Series 300, Series 400, and Series 700. It gives an overview of how to use VUE; it does not cover any specific topics in depth.

Motivation

Because HP VUE is a simple, graphical user interface, it makes HP-UX system workstations much easier to operate. The environment is window-based and mouse driven with a simple, flexible set of utilities.

Exercises

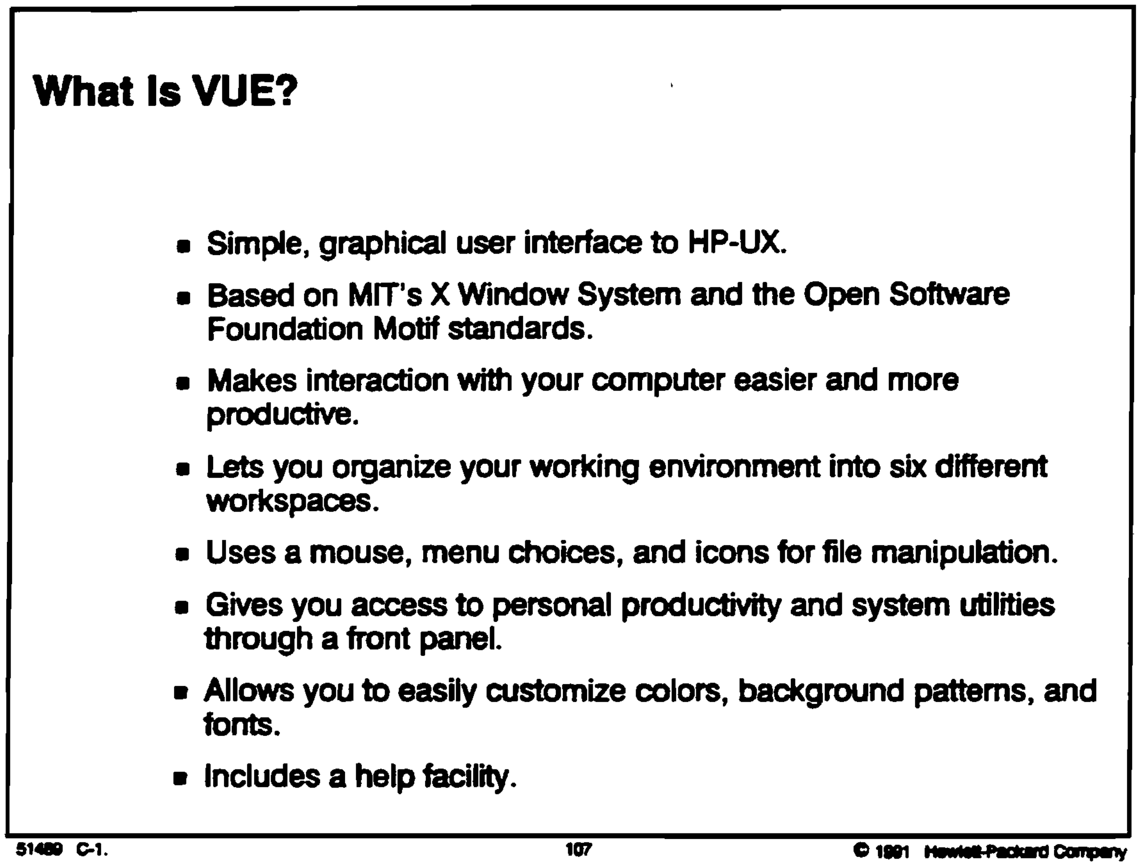
There are exercises throughout this module. Students are encouraged to play with VUE after the module to reinforce the skills learned.

Module Transition

What optional module you discuss next is your choice.

Appendix C — HP Visual User Environment (VUE)

C-1. SLIDE: What Is VUE?



What Is VUE?

- Simple, graphical user interface to HP-UX.
- Based on MIT's X Window System and the Open Software Foundation Motif standards.
- Makes interaction with your computer easier and more productive.
- Lets you organize your working environment into six different workspaces.
- Uses a mouse, menu choices, and icons for file manipulation.
- Gives you access to personal productivity and system utilities through a front panel.
- Allows you to easily customize colors, background patterns, and fonts.
- Includes a help facility.

51489 C-1. 107 © 1991 Hewlett-Packard Company

Student Notes

HP VUE is a simple, graphical user interface to HP-UX designed to make interaction with your computer easier and more productive. It provides a graphical environment featuring window-based, mouse driven functionality, and a simple, flexible set of utilities. In this environment you can have a collection of windows and icons that allow you to start several programs simultaneously and interact with all of them. In addition, you may have multiple workspaces, each with its own collection of windows and icons. It is like having six monitors on your desk!

HP VUE is based on MIT's X Window System and the Open Software Foundation Motif standards.

Appendix C — HP Visual User Environment (VUE)

In this module we will discuss the basic functionality of HP VUE including the following:

- Using the mouse.
- Using the Workspace Manager.
- Starting a terminal window.
- Working with windows.
- Using the File Manager.
- Using the Help Manager.
- Using the Style Manager.

HP VUE requires a bit-mapped display and some type of pointing device, such as a mouse.

C-1. SLIDE: What Is VUE?

Purpose

To introduce HP VUE and list what will be covered in the module.

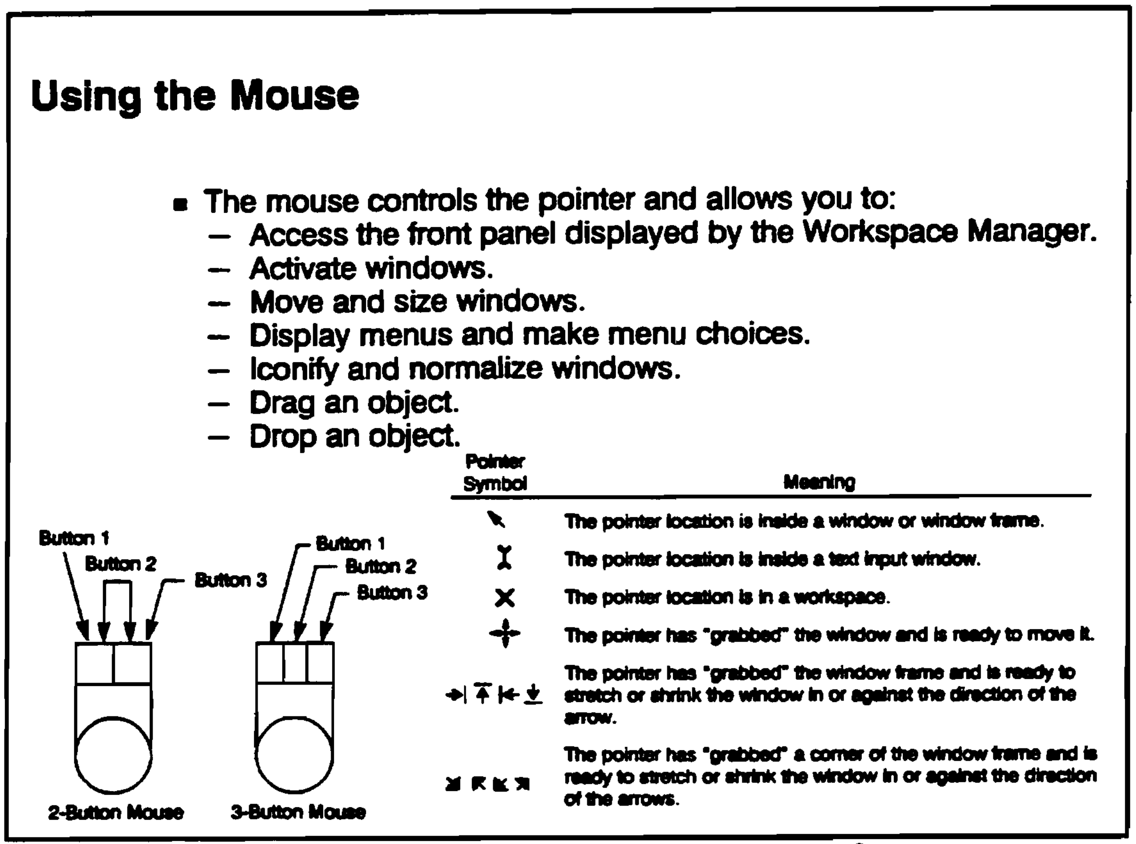
Key Points

Since HP VUE is based on the X Window System, it provides the "windows" on the display that allow you to work with many processes simultaneously, and it allows you to establish sessions on remote computers through a LAN while maintaining your local working environment. In addition, it offers a graphical user environment that shields you from UNIX operating system commands. Instead of commands, you use a mouse, menu choices, and icons to complete many tasks.

Transition

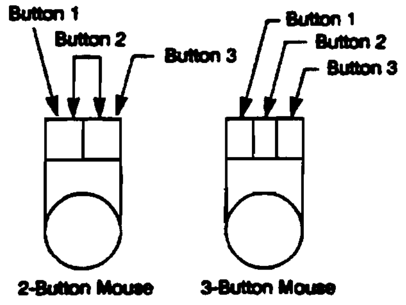
We use the mouse to control the HP VUE environment.

C-2. SLIDE: Using the Mouse



Using the Mouse

- The mouse controls the pointer and allows you to:
  - Access the front panel displayed by the Workspace Manager.
  - Activate windows.
  - Move and size windows.
  - Display menus and make menu choices.
  - Iconify and normalize windows.
  - Drag an object.
  - Drop an object.



2-Button Mouse 3-Button Mouse

Pointer Symbol	Meaning
↖	The pointer location is inside a window or window frame.
I	The pointer location is inside a text input window.
✕	The pointer location is in a workspace.
✣	The pointer has "grabbed" the window and is ready to move it.
→| ⤒ |← ⤓	The pointer has "grabbed" the window frame and is ready to stretch or shrink the window in or against the direction of the arrow.
↘ ↖ ↙ ↗	The pointer has "grabbed" a corner of the window frame and is ready to stretch or shrink the window in or against the direction of the arrows.

51489 C-2. 108 © 1991 Hewlett-Packard Company

Student Notes

The **mouse** is a pointing device that lets you control what part of the system you will interact with. It controls the pointer on your screen. We will use the mouse to do the following:

- Access the front panel displayed by the Workspace Manager.
- Activate windows.
- Move and size windows.
- Display menus and make menu choices.
- Iconify and normalize windows.
- Drag an object.
- Drop an object.

Each of these mouse actions will be discussed as needed. Remember the mouse is how we interact with HP VUE.

Most mice either have two or three buttons. We use these buttons to indicate what we want to do with the item that we are pointing to. Throughout this module we will refer to the "left button", the "right button", and the "middle button". On a two-button mouse, the "middle button" means you press the *left and right* buttons simultaneously.

To **click** a button means to press and then release the indicated button (usually the left button). To **double-click** a button means to press and then release the indicated button twice quickly (usually the left button). To **move the pointer** means to move the mouse. To **drag an object** means to move the pointer over the object, press and hold down the mouse button (usually the left button), move the mouse, and release the button. To **drop** an object means to move the pointer over the object, press and hold down the mouse button (usually the middle button), move the mouse, and release the button.

The pointer on your screen has several meanings depending on where it is pointing. The shape of the pointer will change to indicate what you can do. The different pointer shapes and functions are listed on the slide.

C-2. SLIDE: Using the Mouse

Instructor Notes

Purpose

HP VUE is very difficult to use without a mouse. This describes how we will refer to the mouse throughout the chapter.

Key Points

All HP mice have either two or three buttons. HP VUE uses three buttons: left, middle, and right. On a two-button mouse, the middle button is the left and right buttons together.

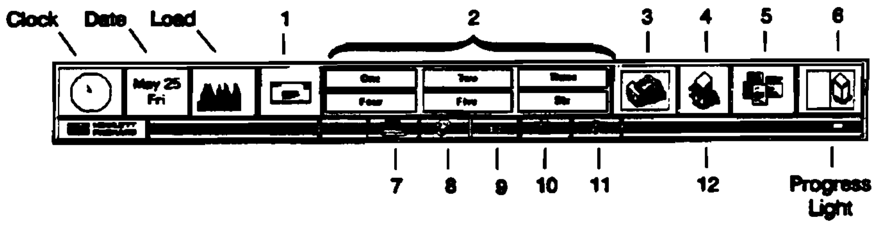
Transition

When you first get into HP VUE, you are placed in the first workspace. Let's see what we can do in this workspace.

C-3. SLIDE: Using the Workspace Manager

Using the Workspace Manager

- Provides a central location (front panel) for indicators and frequently used functions.
- Lets you manage up to six separate workspaces or screens.



51469 C-3. 109 © 1991 Hewlett-Packard Company

Student Notes

The Workspace Manager has a front panel with indicators that display current information about your workstation. The panel also contains a selection of services to give you fast access to personal productivity and system utilities.

The indicators (labeled in the slide) are:

Clock	Displays the current workstation time.
Date	Displays the current workstation date.
Load	Indicates how hard your workstation is working.
Progress Light	Blinks to indicate an activity in progress.

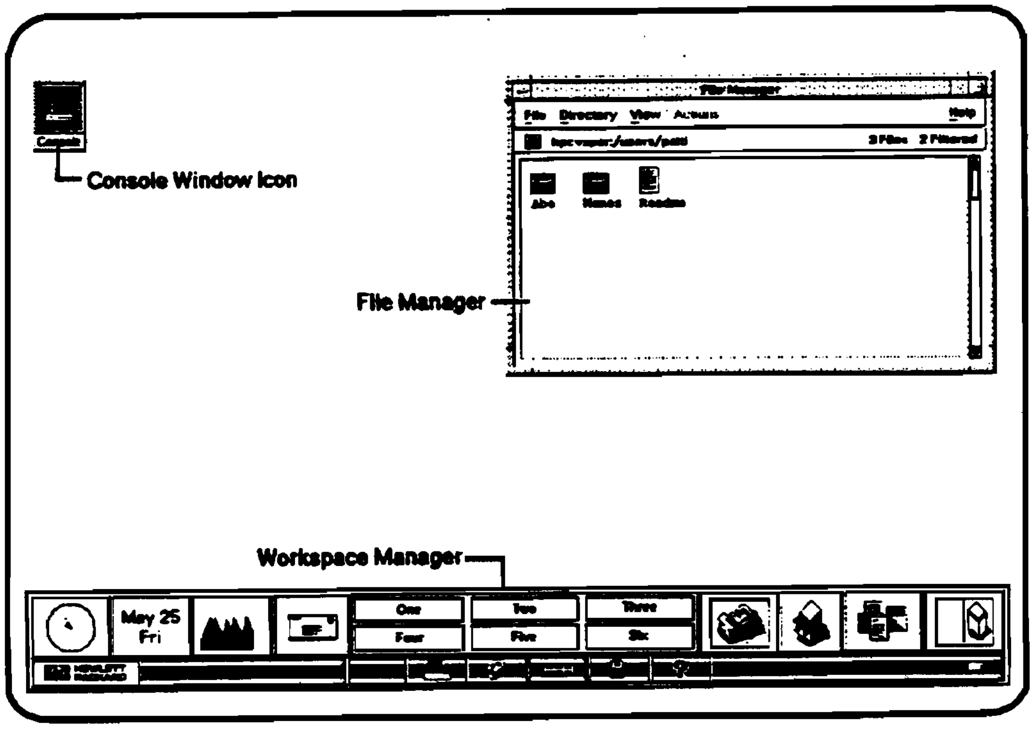
Clicking the progress light starts the logout process. Clicking the other indicators has no meaning.

The available services (numbered in the slide) are:

1. Mail	Displays a dialog box for specifying a mail destination when you drop a file icon on the mail icon. When clicked, it starts the `elm` electronic mail program. The mail icon will change when new mail arrives.
2. Workspace Switch	Switches workspaces.
3. Print	Prints a file when you drop that file icon on the print icon. When clicked, it opens a window that shows the status of the printers.
4. File Manager	Starts a file manager view of your home directory.
5. Applications Directory	Starts a file manager view of applications available on your workstation.
6. Trash Can	Removes a file when you drop the file icon on the trash icon. When clicked, it opens a window that displays the contents of your "trash can".
7. Terminal Window	Starts a terminal window, providing access to the command-line prompt.
8. Style Manager	Starts the style manager.
9. Rename Workspace	Displays the dialog box you use to rename your workspaces.
10. Lock	Locks your workstation screen, preventing keyboard and mouse input. Your password is required to unlock your session.
11. Help Manager	Starts the help manager.
12. Logout Button	Begins the logout process. The progress light is part oft his button.

In addition to these facilities, the Workspace Manager lets you manage multiple workspaces. Each workspace is a separate screen or collection of windows and icons. It is like having six monitors on your desk! Multiple workspaces allow you to organize your work in the most meaningful way, creating different workspaces for different tasks. These multiple workspaces help you to avoid "window clutter".

Your first workspace should look similar to this:



Exercises

1. Start two terminal windows in your current workspace.

2. Switch to another workspace and start a terminal window there.

3. Toggle between your two workspaces.

4. Using the Mail icon on the front panel, mail your partner one of your files.

5. Print a file in your home directory by using the Printer icon.

C-3. SLIDE: Using the Workspace Manager — Instructor Notes

Purpose

The Workspace Manager gives the user easy access to many system utilities. In addition, it manages the different workspaces so that the user can set up different "monitors" for different tasks.

Key Points

The front panel gives you easy access to the many tasks that occur during every work session. Also, the idea of having essentially six different monitors appeals to many students.

You may also point out that you can recover a "trashed" file by clicking the trash can, clicking the file you want to restore, and choosing `Restore` from the Trash Can window's `Edit` menu.

Transition

Let's see how we can work in a terminal window.

Answers

1. Start two terminal windows in your current workspace.

Answer:

Click twice on the terminal window on the front panel.

2. Switch to another workspace and start a terminal window there.

Answer:

Click on another workspace, then click on the terminal window.

3. Toggle between your two workspaces.

Answer:

Click on the two workspaces you have activated.

4. Using the Mail icon on the front panel, mail your partner one of your files.

Answer:

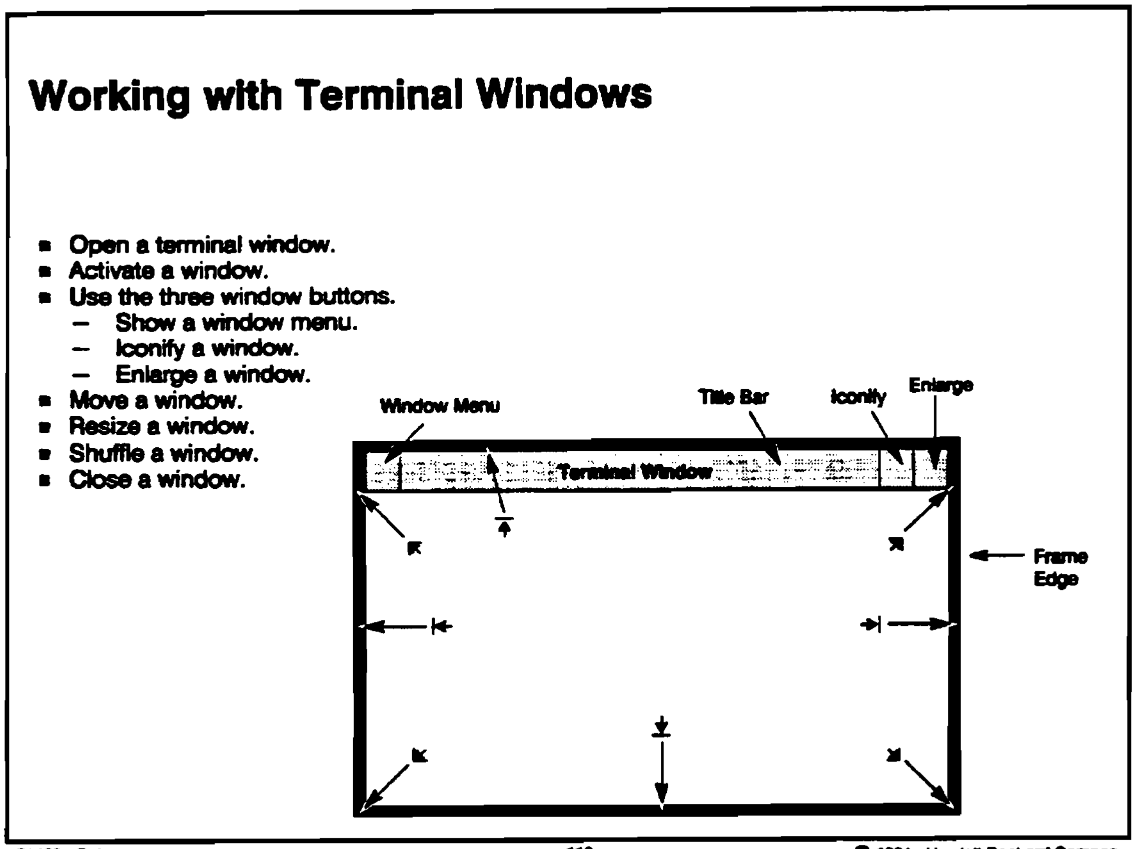
Press and hold down the middle mouse button on a file in the file manager, drag the file over the mail icon, release the middle mouse button. Position the pointer in the displayed dialog box so that you can type the destination and an optional subject, press Return.

5. Print a file in your home directory by using the Printer icon.

Answer:

Drag a file from the file manager down to the print icon.

C-4. SLIDE: Working with Terminal Windows



Working with Terminal Windows

- Open a terminal window.
- Activate a window.
- Use the three window buttons.
  - Show a window menu.
  - Iconify a window.
  - Enlarge a window.
- Move a window.
- Resize a window.
- Shuffle a window.
- Close a window.

Window Menu
Title Bar
Iconify
Enlarge
Terminal Window
Frame Edge

51480 C-4. 110 © 1991 Hewlett-Packard Company

Student Notes

To open a terminal window, click the terminal button on the front panel. Terminal windows provide access to the command line prompt, and through it, operating system commands. This terminal window is called an **hpterm** window; it emulates a Hewlett-Packard text terminal.

Although you may have many windows in your workspace, only one may be active (receiving keyboard input) at a time. To **activate** a window, move the mouse so that the pointer is anywhere on the window, and click the left button. You should see the color of the window frame change. This technique is also used to shuffle windows. If you click on a concealed window, the window will be shuffled to the top, and the color of the window frame will change.

Notice the different parts of the window frame:

Title bar	This is the name of the window. It is also used for moving the window—move the pointer over the title bar, *press and hold* down the left mouse button, drag the window to a new location, release the mouse button.

Window menu	If you *press* the left mouse button on this part of the frame, a menu will appear for the window. To make a selection, move the pointer and click on your menu choice. You can also activate the window menu by moving the pointer to any point on the window frame and pressing and holding the right button. Double-clicking on the window menu button will close the window.
Iconify	This "button" allows you to turn your window into a small graphical image (an icon) to help unclutter your screen.
Enlarge	This "button" will make the window take up the entire screen. It will cover the whole work space and all other windows.
Frame edge	This is used to resize the window—move the pointer to the edge of the frame, *press and hold* down the left mouse button, drag window frame to new size, release the mouse button. Moving the pointer to a *corner* of the frame will resize the frame in both dimensions.

The Window menu controls the window. Notice that nearly every option on the Window menu can also be accomplished using a part of the window frame. One Window menu item you cannot perform from the frame is the `Restore` option. This will either normalize an iconified window or restore a maximized window to its normal size. (Double-clicking on an icon will also restore the icon to its normal size.) Another Window menu choice is the `Close` option. This will kill the window, stopping the process that was running in it.

Exercises

1. Create two terminal windows in your workspace.

2. Move one of the terminal windows to a different location in your workspace.

3. Iconify the other terminal window and then restore it to its normal size.

4. Make one of the windows bigger using the mouse and frame edge.

5. Activate the other terminal window and execute the `ls -l` command to see that it works just like a text terminal.

6. Close one of the windows.

C-4. SLIDE: Working with Terminal Windows — Instructor Notes

Purpose

HP VUE terminal windows provide access to the command line prompt. We also see from this slide how to manipulate windows using the window frame.

Key Points

Terminal windows act just like text terminals. `hpterm` is a program that starts a window that emulates an HP terminal.

You must click the left button on a window to make it active.

Most window manipulation functions can be performed either from the Window menu or from the window frame.

You can activate the Window menu by pressing the left button on the Window menu box *or* by pressing and holding the right button on any part of the frame.

Where Problems Arise

When you resize a terminal window, applications running in the window may not realize that the window has been resized. To inform the contents of a terminal window that the window is a different size, enter the following:

```
$ eval 'resize'
```

In HP VUE, personalized environment variables are kept in the `.vueprofile` file in your home directory. Therefore, if you want to use the `vi` editor to recall and edit your previous commands (assuming you are in the Korn Shell), you must add the following to your `.vueprofile`:

```
EDITOR=/usr/bin/vi
export EDITOR
```

Point out that all processes running in a window should be stopped before closing that window. If you are in `vi` when the window is closed, the editor will be killed.

Transition

Let's see what information is contained in the file manager window.

Answers

1. Create two terminal windows in your workspace.

Answer:

Click twice on the terminal button on the front panel.

2. Move one of the terminal windows to a different location in your workspace.

Answer:

Move the pointer to the title bar, press and hold the left button, drag the window to a new location.

3. Iconify the other terminal window and then restore it to its normal size.

Answer:

Click on the iconify button. To restore the icon to its normal size, double click on the icon, or click the left button on the icon to bring up the window menu, and choose `Restore`.

4. Make one of the windows bigger using the mouse and frame edge.

Answer:

Move the pointer to a place on the window frame, press and hold down the left button, drag the frame to its new size, and release the button.

5. Activate the other terminal window and execute the `ls -l` command to see that it works just like a text terminal.

Answer:

Click on the other window and execute `ls -l`.

6. Close one of the windows.

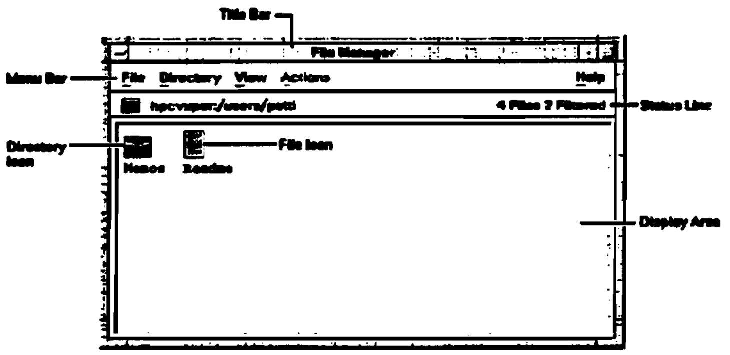
Answer:

Double click on the window menu button.

C-5. SLIDE: The File Manager

The File Manager

- Organizes and manages information stored in the workstation.
- File and directory management actions controlled from menu bar.
  - File menu
  - Directory menu
  - View menu
  - Actions menu
  - Help menu
- Status line provides information on current location.
- Display area shows contents of current directory.



51489 C-5. 111 © 1991 Hewlett-Packard Company

Student Notes

The HP VUE file manager helps you organize and manage the information stored in your workstation. You can always get a file manager view by clicking on the file manager button on the front panel; the file manager will display your home directory. Data file icons are sheets of paper; directory icons are folders; the icon for an executable file is a lightning bolt. You can create a new file manager window for a directory by dropping the directory icon on the workspace background.

The menu bar provides five pull down menus:

`File` menu	Contains actions for creating, finding, and removing files.
`Directory` menu	Contains actions applicable to creating, changing, and viewing directories.
`View` menu	Contains actions that affect your view of the files and directories contained in a file manager display area.
`Actions` menu	Contents are context-sensitive and change to reflect possible actions for the file or directory selected.
`Help` menu	Provides help on the file manager.

The status line contains the **host name** of the workstation, the **path** of the directory being viewed, and the number of files and **filtered** files that the current view contains. A **filtered** file does not appear in the display area; these are usually the "dot" files. Double-clicking a path segment will display that directory's contents in the display area.

If the current directory contains more files than fit within the display area, a scroll bar appears to the right of the area. The scroll bar allows you to scroll the window to view its entire contents. Simply press and hold the left mouse button and drag the slider.

Exercises

1. Create a new file manager window for the directory **cars+dogs** that is in your current file manager window.

2. Display the contents of your home directory from the new file manager you created in exercise 1.

C-5. SLIDE: The File Manager

Instructor Notes

Purpose

We need to know what is contained in a file manager window.

Key Points

Most file and directory management actions are accessed through the file manager menu bar. You can do some tasks by dragging a file or directory icon and dropping it in a particular location. Other tasks require you to select a file or directory first. These tasks are grayed out in the menus until a selection is made.

Transition

Let's see how the file manager can be used to manipulate files.

Answers

1. Create a new file manager window for the directory **cars+dogs** that is in your current file manager window.

Answer:

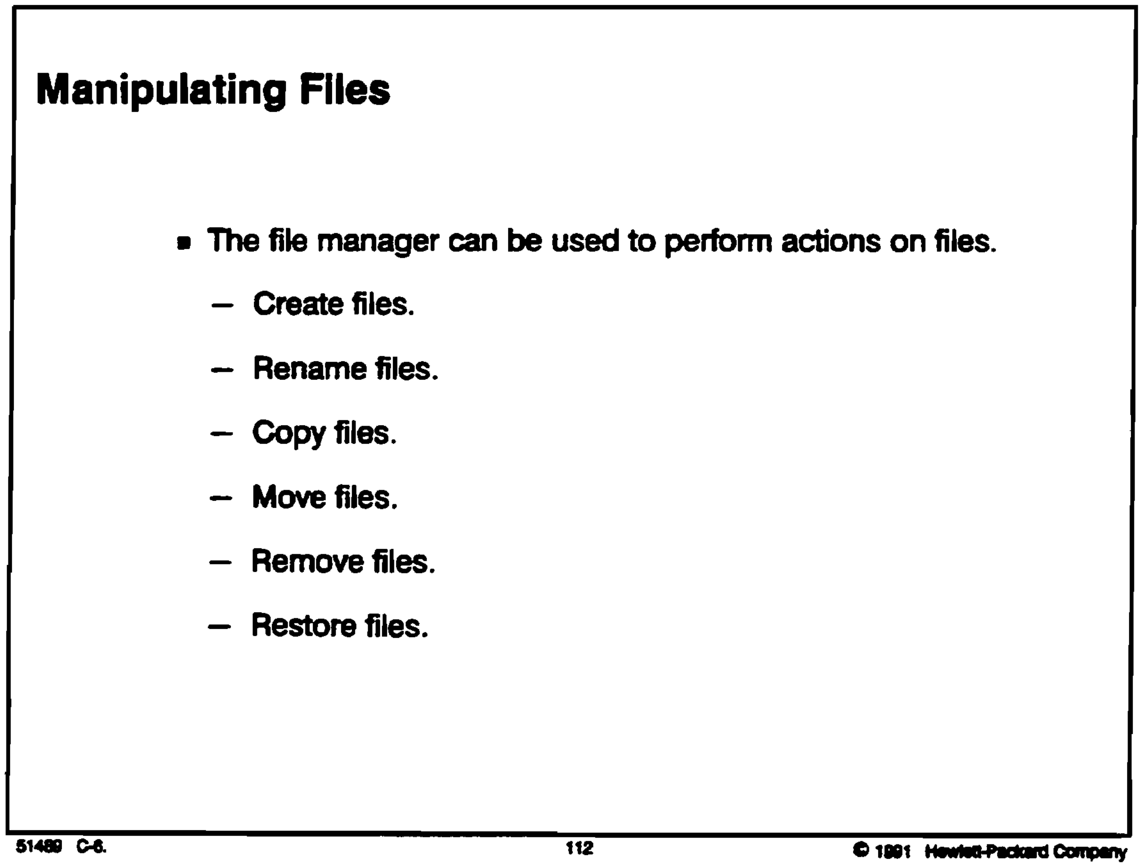
Move the pointer over the directory icon for **cars+dogs**. Press and hold down the middle mouse button, drag the icon over the workspace background, and release the middle mouse button.

2. Display the contents of your home directory from the new file manager you created in exercise 1.

Answer:

Double-click your home directory path segment on the status line in the file manager for **cars+dogs**.

C-6. SLIDE: Manipulating Files



Manipulating Files

- The file manager can be used to perform actions on files.
  - Create files.
  - Rename files.
  - Copy files.
  - Move files.
  - Remove files.
  - Restore files.

51489 C-6. 112 © 1991 Hewlett-Packard Company

Student Notes

To create a file, choose `New ...` from the `File` menu to open the `New Data File` dialog box. Click the `New Data File Name` box, type in the name of the file you wish to create, press Return.

To rename a file, click the file whose name you wish to change. Choose `Rename ...` from the `File` menu to open the `Rename File` dialog box. Double-click the `New File Name` box over an empty area, type in the new name, press Return.

To copy a file, move the pointer over the file you wish to copy, press and hold down Ctrl, press and hold down the middle mouse button. Drag the file over an empty area in another file manager which displays the directory to which you are copying, release the middle mouse button, release Ctrl.

To move a file, move the pointer over the file you wish to move, press and hold down the middle mouse button. Drag the file over an empty area in another file manager which displays the directory to which you are moving the file, release the middle mouse button.

To remove a file, press and hold down the middle mouse button over the file you wish to remove. Drag the file over the trash can, release the middle mouse button.

To restore a file, click the trash can in the workspace manager to open the `Trash Can` window, click the name of the file you wish to restore. Choose `Restore` from the Trash Can window's `Edit` menu. Choose `Exit` from the Trash Can window's `File` menu.

Exercises

1. Create files `new.one` and `new.two` in the directory `cars+dogs`.

2. Rename `new.one` to `new.old`.

3. Copy `new.two` to your parent directory.

4. Move `new.old` to your parent directory.

5. Remove `new.two` and `new.old` from your parent directory.

6. Restore `new.old` to your parent directory.

C-6. SLIDE: Manipulating Files

Instructor Notes

Purpose

The VUE File Manager provides a relatively easy way for users to create, rename, copy, move, remove, and restore files without knowing the HP-UX commands.

Key Points

Some file manipulation tasks require you to drag a file icon and drop it in a particular location. Other tasks require you to use the File Manager menu bar.

Where Problems Arise

When renaming a file, make sure to double-click on the `New File Name` box. A single click will put you in *append* mode.

When copying a file, the middle mouse button must be released *before* (Ctrl). Releasing (Ctrl) first will do a *move*.

You can restore files for a limited time only. They will be *permanently* removed when you log out.

Transition

Let's see how the file manager can be used to manipulate directories.

Answers

1. Create files `new.one` and `new.two` in the directory `cars+dogs`.

Answer:

Double click on `cars+dogs` to open a file manager view of this directory. Choose `New ...` from the `File` menu, click the `New Data File Name` box, type `new.one`, press (Return). Repeat the process for `new.two`.

2. Rename `new.one` to `new.old`.

Answer:

Click `new.one`, choose `Rename ...` from the `File` menu, position pointer at end of `new.one` in the `New File Name` box, click, back space as necessary, type `new.old`, press (Return).

3. Copy `new.two` to your parent directory.

Answer:

Click the file manager button in the workspace manager to open another File Manager view of your home (parent) directory. Move your pointer over `new.two`, press and hold down Ctrl, press and hold down the middle mouse button. Drag `new.two` over an empty area in the parent directory, release the middle mouse button, and release Ctrl.

4. Move `new.old` to your parent directory.

Answer:

Press and hold down the middle mouse button on `new.old`, drag `new.old` over an empty area in the parent directory, and release the middle mouse button.

5. Remove `new.two` and `new.old` from your parent directory.

Answer:

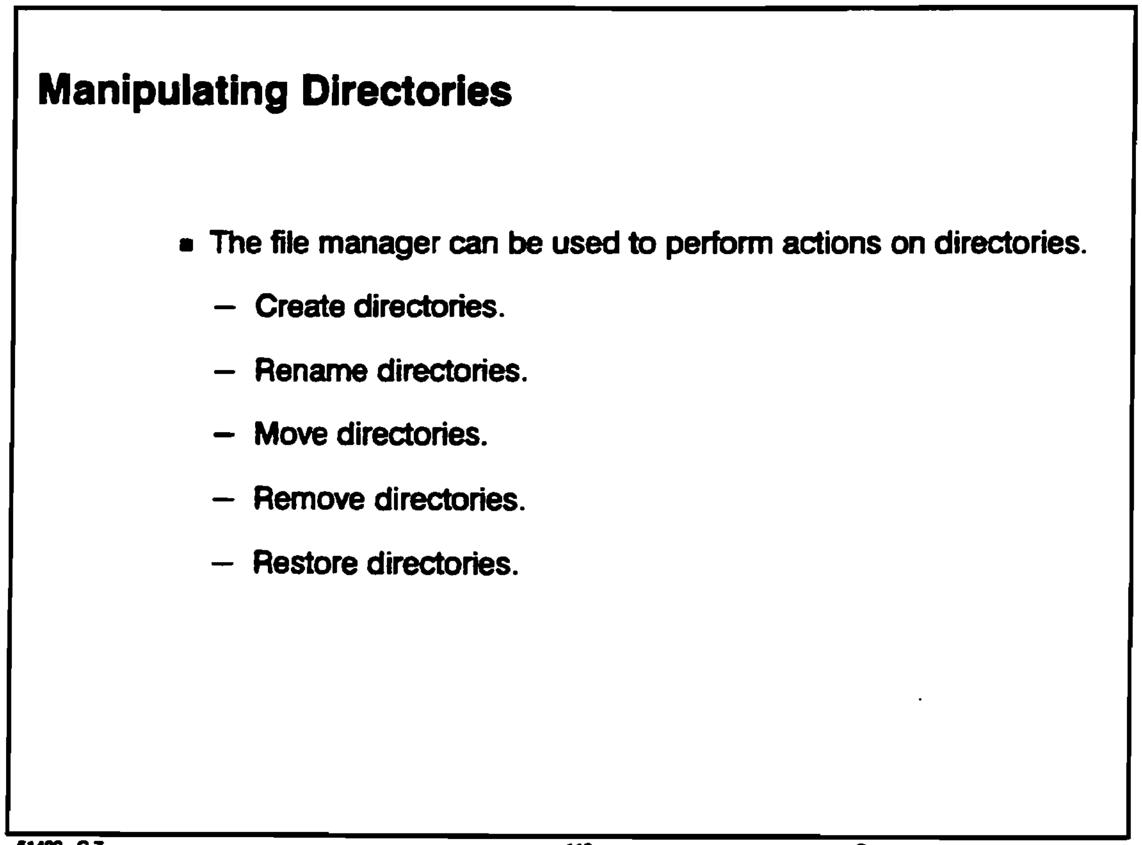
Press and hold down the middle mouse button on `new.two` in the parent directory, drag the file over the trash can, and release the middle mouse button. Repeat the process for `new.old`.

6. Restore `new.old` to your parent directory.

Answer:

Click the Trash Can in the workspace manager, click `new.old` in the Trash Can list area, choose `Restore` from the `Edit` menu, and choose `Exit` from the `File` menu.

C-7. SLIDE: Manipulating Directories



Manipulating Directories

- The file manager can be used to perform actions on directories.
  - Create directories.
  - Rename directories.
  - Move directories.
  - Remove directories.
  - Restore directories.

51489 C-7. 113 © 1991 Hewlett-Packard Company

Student Notes

To create a directory, choose `New ...` from the `Directory` menu to open the `New Directory` dialog box. Click the `New Directory Name` box, type in the name of the directory you wish to create, press (Return).

To rename a directory, click the directory whose name you wish to change. Choose `Rename ...` from the `File` menu to open the `Rename File` dialog box. Double-click the `New File Name` box over an empty area, type in the new name, press (Return).

To move a directory, move the pointer over the directory you wish to move, press and hold down the middle mouse button. Drag the directory over an empty area in the file manager to which you are moving the directory, release the middle mouse button.

To remove a directory, press and hold down the middle mouse button over the directory you wish to remove. Drag the file over the trash can, release the middle mouse button.

To restore a directory, click the trash can in the workspace manager to open the `Trash Can` window, click the name of the directory you wish to restore. Choose `Restore` from the Trash Can window's `Edit` menu. Choose `Exit` from the Trash Can window's `File` menu.

Exercises

1. Create directories `DIRECT` and `SHOW` in the directory `cars+dogs`.

2. Rename `DIRECT` to `REDIRECT`.

3. Move `SHOW` to your parent directory.

4. Remove `SHOW` from your parent directory.

5. Restore `SHOW` to your parent directory.

C-7. SLIDE: Manipulating Directories

Instructor Notes

Purpose

The VUE File Manager provides a relatively easy way for users to create, rename, move, remove, and restore directories without knowing the HP-UX commands.

Key Points

Some directory manipulation tasks require you to drag a directory icon and drop it in a particular location. Other tasks require you to use the File Manager menu bar.

Where Problems Arise

When renaming a directory, make sure to double-click on the `New File Name` box. A single click will put you in *append* mode.

You can restore directories for a limited time only. They will be *permanently* removed when you log out.

Transition

Let's see how to use the Help Manager.

Answers

1. Create directories `DIRECT` and `SHOW` in the directory `cars+dogs`.

Answer:

Choose `New ...` from the `Directory` menu, click the `New directory Name` box, type `DIRECT`, press Return. Repeat the process for `SHOW`.

2. Rename `DIRECT` to `REDIRECT`.

Answer:

Click `DIRECT`, choose `Rename ...` from the `File` menu, position pointer at beginning of `DIRECT` in the `New File Name` box, click, type `REDIRECT`, and press Return.

3. Move `SHOW` to your parent directory.

Answer:

Press and hold down the middle mouse button on SHOW, drag SHOW over an empty area in the parent directory, and release the middle mouse button.

4. Remove SHOW from your parent directory.

Answer:

Press and hold down the middle mouse button on SHOW in the parent directory, drag the file over the trash can, and release the middle mouse button.

5. Restore SHOW to your parent directory.

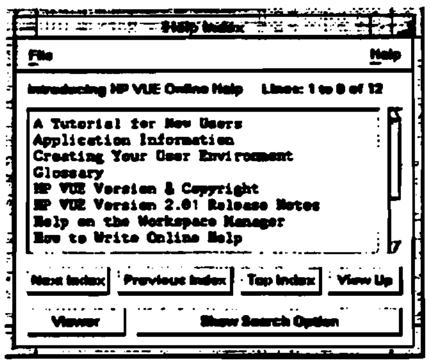
Answer:

Click the Trash Can in the workspace manager, click SHOW in the Trash Can list area, choose `Restore` from the `Edit` menu, and choose `Exit` from the `File` menu.

C-8. SLIDE: Using the Help Manager

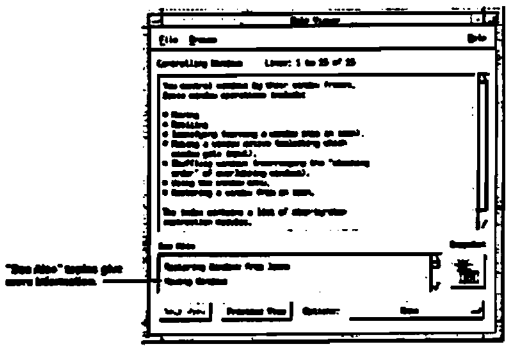
Using the Help Manager

- Help is available for any HP VUE application.
  - Ask the workspace manager for the help index.
  - Choose `On Application...` from an application's `Help` menu.
  - Click a Help button in a dialog box.



Introducing HP VUE Online Help   Lines: 1 to 8 of 12

```
A Tutorial for New Users
Application Information
Creating Your User Environment
Glossary
HP VUE Version & Copyright
HP VUE Version 2.01 Release Notes
Help on the Workspace Manager
How to Write Online Help
```



51489 C-8.   114   © 1991 Hewlett-Packard Company

Student Notes

The Help Manager is a system-wide, context-sensitive help facility. It provides help on a variety of topics and applications. You may explore any topic in as much detail as you want and in any order.

To get an index to topics for which you can get on-line help, click the help button, ?, on the front panel of the workspace manager. When the `Help Index` window appears, click a topic. In most cases, you will get another `Help Index` window on this subtopic. You may either choose another subtopic or click Viewer to view information about the topic already chosen; in either case, the `Help Viewer` window will be displayed. Use the scroll bar as necessary to view the contents of any window. A `Help Viewer` window or `Help Index` window may be closed using the `File` menu.

Each window shows the topic name, the lines displayed, and the total number of lines. This information appears in the status line above the window's display area.

The main window of each HP VUE application has a `Help` menu on its menu bar. Choose `On Application...` from the `Help` menu. A `Help Viewer` window will appear that will display general information about that particular application.

When you click (Help) in a dialog box, you will receive general help about that dialog box; a `Help Viewer` window will be displayed.

Exercises

1. Open the `Help Index` window from the front panel of the workspace manager; display information on the topic `Help on the Workspace Manager`.

2. Go to a `File Manager` window and display the `Help Viewer` window for this application.

3. Go to a `File Manager` window and bring up a dialog box; get help on this dialog box.

C-8. SLIDE: Using the Help Manager

Instructor Notes

Purpose

The Help Manager provides on-line access to information about many topics.

Key Points

The Help Manager allows you to explore any topic in as much detail as you want, and in any order. A help index shows related topics, and allows you to quickly choose another topic.

Where Problems Arise

Always check the title bar of the window you are viewing to determine whether you are in `Help Index` or `Help Viewer`.

Choosing some topics from a `Help Index` will cause another `Help Index` to be displayed with additional topics. This can continue for several layers. If there are no additional subtopics, choosing a topic will automatically display a `Help Viewer` window.

To view a topic named in the status line of a `Help Index`, click on `Viewer`. However, if a subtopic is highlighted, you will get a `Help Viewer` window for the highlighted topic. To cancel the highlighting, click on `Previous Index`.

Transition

Let's see how we can customize the environment

Answers

1. Open the `Help Index` window from the front panel of the workspace manager; display information on the topic `Help on the Workspace Manager`.

Answer:

Click the help button, ?, on the front panel of the workspace manager, choose `Help on the Workspace Manager` in the `Help Index`, and click `Viewer` from the `Help Index`.

2. Go to a `File Manager` window and display the `Help Viewer` window for this application.

Answer:

Click `Help` in the file manager menu bar,and choose `On Application` ... .

3. Go to a `File Manager` window and bring up a dialog box; get help on this dialog box.

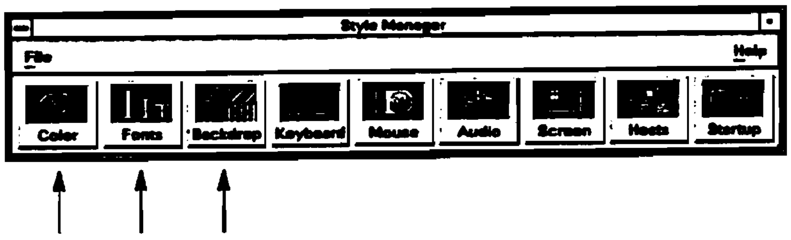
Answer:

Click `File` in the file manager menu bar, choose `new` ... , and click `Help` in the dialog box.

C-9. SLIDE: Using the Style Manager

Using the Style Manager

- Allows you to customize your environment.
  - Colors
  - Background patterns
  - Fonts



51489 C-9. 115 © 1991 Hewlett-Packard Company

Student Notes

The Style Manager allows you to dynamically customize colors, background patterns, and fonts.

The Style Manager is opened from the workspace manager by moving the pointer to the style manager button and clicking the left mouse button. The style manager has nine push buttons that are used in customizing your environment; these push buttons open dialog boxes.

You can change workspace colors by selecting a palette in the Color dialog box. The selected palette is used for all workspaces. Click the Color button in the style manager to open the Color dialog box, choose a palette to apply it to your workspace, click (OK) to close the dialog box.

You can choose a backdrop for your workspace from a list of patterns in the Backdrop dialog box. To make a workspace easy to recognize, choose a different pattern for each workspace. You must be *in* the workspace to change its backdrop pattern. Click the Backdrop button in the style manager to open the Backdrop dialog box, choose a backdrop name to show the pattern in the viewing area, click (Apply) to fill the workspace background with the pattern, click (Close) to close the Backdrop dialog box.

You can choose small, medium, or large fonts. When you choose a font size, you must restart your HP VUE session (log out and log back in) to use the new font size. Click Fonts in the style manager to open the Fonts dialog box, hold down the left mouse button on the `Font Size` option menu button to show the choices, choose `Small`, `Medium`, or `Large`. Click OK in the Notice dialog box that opens after a font size is chosen, click OK in the Fonts dialog box. Select `Resume current session` in the Session Startup dialog box; this box is opened when you click Startup in the style manager. Finally, log out and log back in.

Exercises

1. Change your workspace colors.

2. Change the backdrop pattern of one of your workspaces.

3. Change the font size to large.

C-9. SLIDE: Using the Style Manager — Instructor Notes

Purpose

This slide gives a brief introduction to customizing the HP VUE environment.

Key Points

A novice user can give his/her HP VUE session a customized look with relative ease. The changes that are made to the environment are saved so that the next time you log in, the session will have the same settings as before.

Transition

The next module is your choice.

Answers

1. Change your workspace colors.

Answer:

Click the Color button in the style manager, choose a color, and click (OK).

2. Change the backdrop pattern of one of your workspaces.

Answer:

Click the Backdrop button in the style manager, choose a backdrop, click (Apply), click (Close).

3. Change the font size to large.

Answer:

Click the fonts button in the style manager, choose `Large` from the `Font Size` option menu, click (OK) in the Notice dialog box, click (OK) in the fonts dialog box, select `Resume current session` in the Session Startup dialog box, logout, and log back in.

Appendix D — Using Network Services

Objectives

Upon completion of this module, you will be able to do the following:

- Describe the different network services in HP-UX.
- Explain the function of a Local Area Network (LAN).
- Find the host name of the local system and other systems in the LAN.
- Use the ARPA/Berkeley Services to perform remote logins, remote file transfers, and remote command execution.

Appendix D — Using Network Services

Overview of Module D

It is very common for a user to purchase a computer to use in a networked environment. This module gives the basic forms of some user LAN services. You should take time to suggest that if students are interested in learning more about LAN, they should take the ARPA/Berkeley Services class.

This module does not replace the ARPA/Berkeley Services class! It is just an overview of the basic LAN services. In fact, it is a good introduction to the other LAN classes.

Motivation

Many users have networked environments and would like to know how to use them to become more productive.

Exercises

There are examples throughout the material with exercises at the end of the module.

If you want the students to actually try the exercises, you need a user account on at least two machines. If you only have one machine, LAN is not fun, but it can be done. Just make sure the students are careful about the file names they choose!

You should also make sure each student has a password set so `ftp` will work, and check the `/etc/hosts.equiv` file so the Berkeley Services will work.

Exercise Files

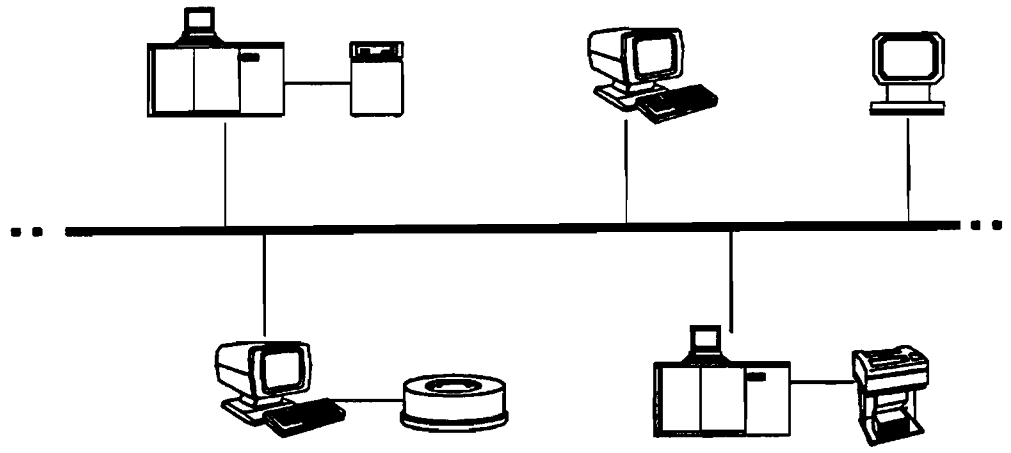
The students can choose the files.

Module Transition

This is usually the last module of the course.

D-1. SLIDE: What Is a Local Area Network?

What Is a Local Area Network?



51489 D-1. 116 © 1991 Hewlett-Packard Company

Student Notes

A **Local Area Network** (LAN) is a method of connecting two or more computer systems over a small area. Most installations that have more than one computer will install a LAN to allow the users to work on several different computers without physically picking up all of their work and moving to the computer they want to work on.

The LAN services that we will be discussing in this module are the programs that allow us to use the LAN to perform many tasks between computers. Some of these tasks are the following:

- Copying files from one computer to another. Without a LAN, you would have to make a tape copy of your files, walk it over to the other computer, and reload the tape.
- Log in to another computer from a terminal on the local computer. Normally you would have to actually go to the other computer to log in.
- Execute commands on another computer and see the results locally. Again, you would have to move to the other computer if you did not have a LAN.

- Access files on a remote computer. This means we will use the files on another computer's disk without copying the files to the local disk.

D-1. SLIDE: What Is a Local Area Network?

Purpose

To introduce the kinds of things we can do with a LAN.

Key Points

These are the four basic functions that users will perform on a LAN. Other functions are normally built out of these four.

Presentation Suggestions

If you have a LAN in the training facility, it helps to discuss the setup of the machines.

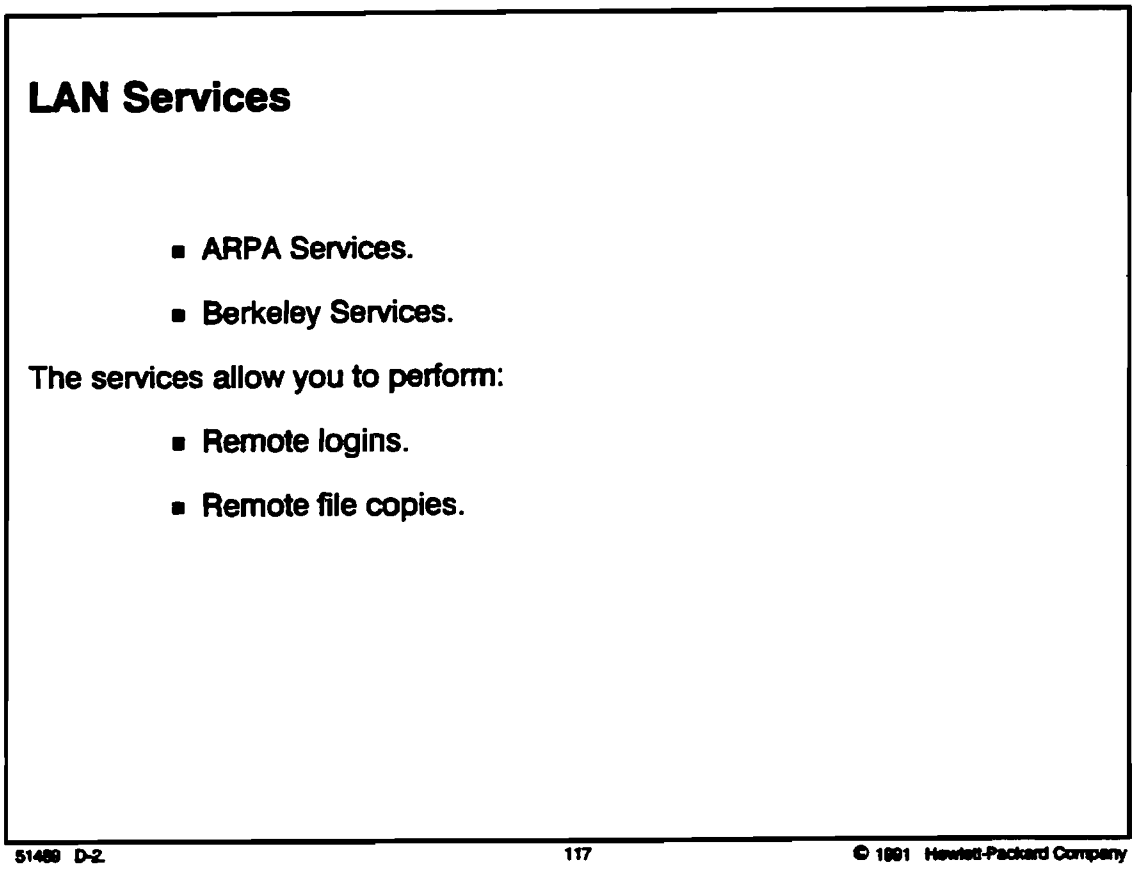
Ask how many students have LAN installations and pick a few to describe why they wanted a LAN in their company. This will help establish the need for a LAN.

Transition

There are two sets of services that you can use to perform these functions:

- ARPA Services.
- Berkeley Services.

D-2. SLIDE: LAN Services



LAN Services

- ARPA Services.
- Berkeley Services.

The services allow you to perform:

- Remote logins.
- Remote file copies.

51489 D-2 117 © 1991 Hewlett-Packard Company

Student Notes

In this module we will look at two different *groups* of services to perform the basic LAN functions we have discussed. These services are the following:

- ARPA Services.
- Berkeley Services.

The ARPA Services were first defined by the Defense Advanced Research Projects Agency (DARPA) in the late 1960s. These services became a standard for communicating to many different brands of computers across a single LAN. The ARPA services that we will discuss are `telnet` and `ftp`.

DARPA hired the University of California at Berkeley and Bold Beranak and Newman (BBN of Massachusetts) to develop these services. In the mid 1970s Berkeley started working with the new UNIX operating system. They eventually developed a more "robust" set of services to be used between computers running the UNIX operating system. These are now called the Berkeley Services. We will introduce the Berkeley services `rcp`, `rlogin`, and `remsh` in this module.

D-2. SLIDE: LAN Services

Instructor Notes

Purpose

We will be showing the many commands that you can use to perform remote tasks. Which commands the students can use will depend on which of these services they have installed on their computers.

Key Points

ARPA Services are used to communicate between many different types of computers with different operating systems. The Berkeley Services are primarily used between UNIX systems (although they do work on other systems).

Where Problems Arise

Many times students have heard about TCP/IP (Transmission Control Protocol/Internet Protocol). This is not a service you can run interactively. Instead, it is the underlying protocol that the network services use to facilitate the communications across the LAN.

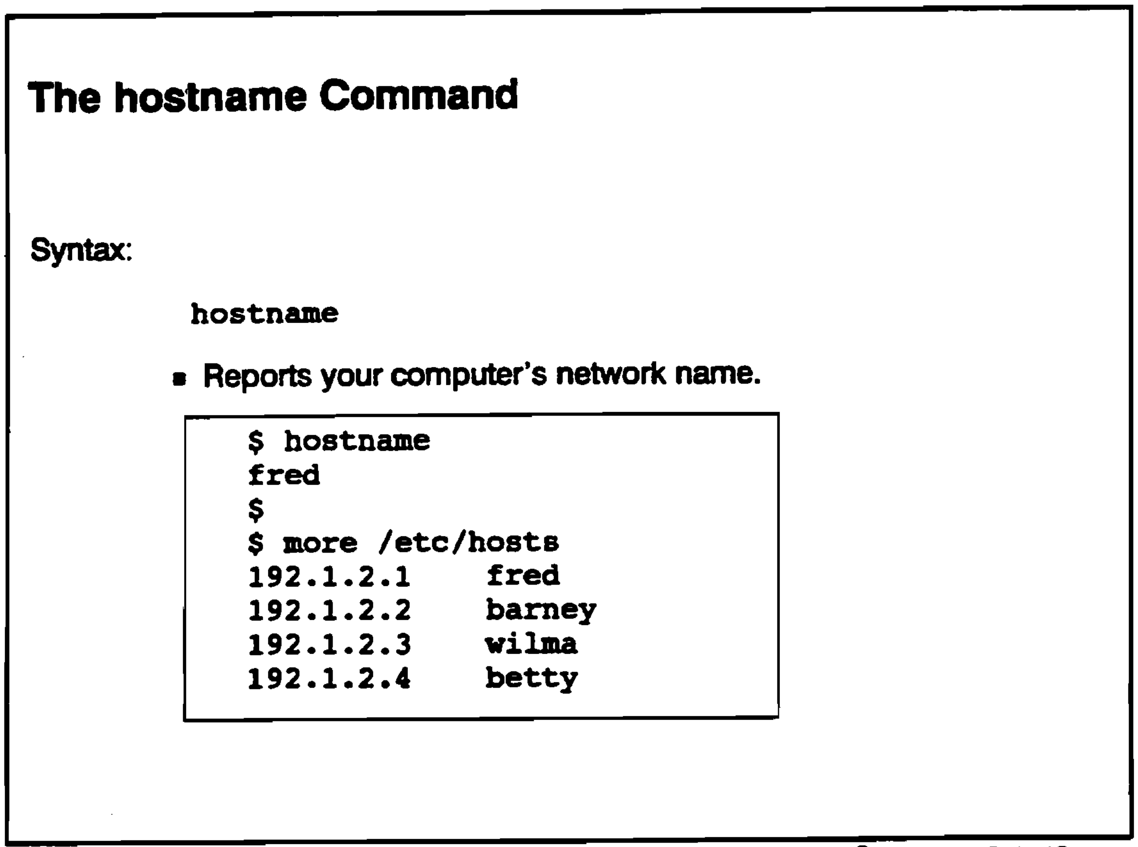
Evaluation Questions

When would you use the ARPA Services or the Berkeley Services? Answer: Use ARPA Services for communicating to computers with different operating systems across a single LAN. Use Berkeley Services for communicating between computers running the UNIX operating system.

Transition

A computer on a LAN is known by its host name.

D-3. SLIDE: The hostname Command



The hostname Command

Syntax:

```
hostname
```

- Reports your computer's network name.

```
$ hostname
fred
$
$ more /etc/hosts
192.1.2.1    fred
192.1.2.2    barney
192.1.2.3    wilma
192.1.2.4    betty
```

51489 D-3. 118 © 1991 Hewlett-Packard Company

Student Notes

Your computer has a host name. This is the name that identifies your system on the LAN. To find your system's host name use the **hostname** command.

```
$ hostname
fred
```

If you want to communicate with another computer on the LAN you must know its host name. You can do this simply by asking the administrator of the other computer what the host name is. You should also check that you have a user account on the machines that you want to work with.

Note In order to use any of the LAN services, *you must be a valid user on the remote computer.*

You can also find host names in the `/etc/hosts` file. However, if you are part of a large LAN installation, this file may contain several hundred entries.

D-3. SLIDE: The hostname Command

Instructor Notes

Purpose

We communicate to other machines using host names.

Key Points

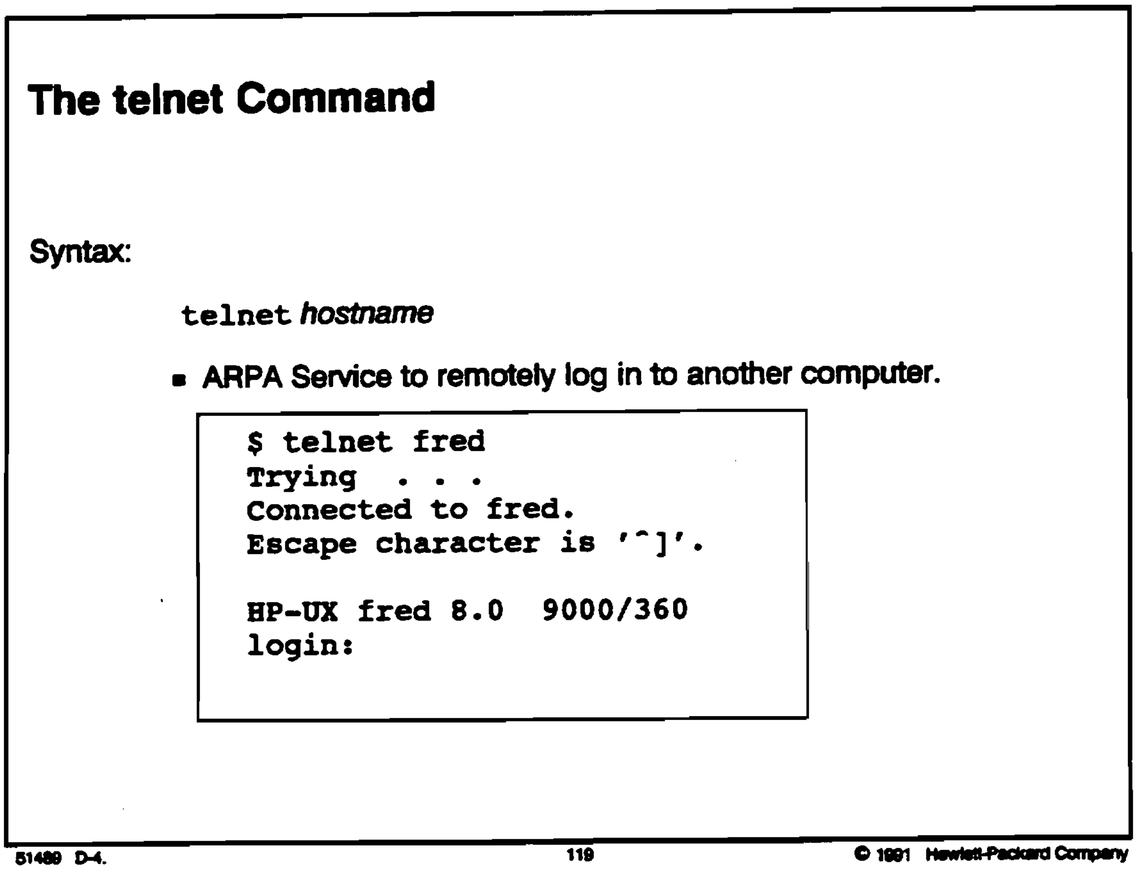
You must be a valid user on the remote computer.

The administrator of the other computer must also allow you access permissions to his or her computer through the network configuration files.

Transition

Let's take a look at the two ARPA Services first.

D-4. SLIDE: The telnet Command



The telnet Command

Syntax:

```
telnet hostname
```

- ARPA Service to remotely log in to another computer.

```
$ telnet fred
Trying . . .
Connected to fred.
Escape character is '^]'.

HP-UX fred 8.0  9000/360
login:
```

51489 D-4. 119 © 1991 Hewlett-Packard Company

Student Notes

`Telnet` is the remote login facility of the ARPA Services.

If you type the command:

```
$ telnet hostname
```

you will see the login prompt for the computer called *hostname* on your screen. At this point, you can enter the user name and password that you use on that machine and you will be logged in.

Once you are logged in, your terminal looks just like it was a terminal on the remote computer. You can run shell commands or programs and even use the remote computer's line printer. *All of the work you do is being executed on the remote computer.* Your local computer is just passing the information to and from your terminal through the LAN.

To close a `telnet` connection, simply log off the remote computer using Ctrl-d Return or `exit`.

D-4. SLIDE: The telnet Command

Instructor Notes

Purpose

To remotely log in to another computer running the ARPA Services, you would use `telnet`.

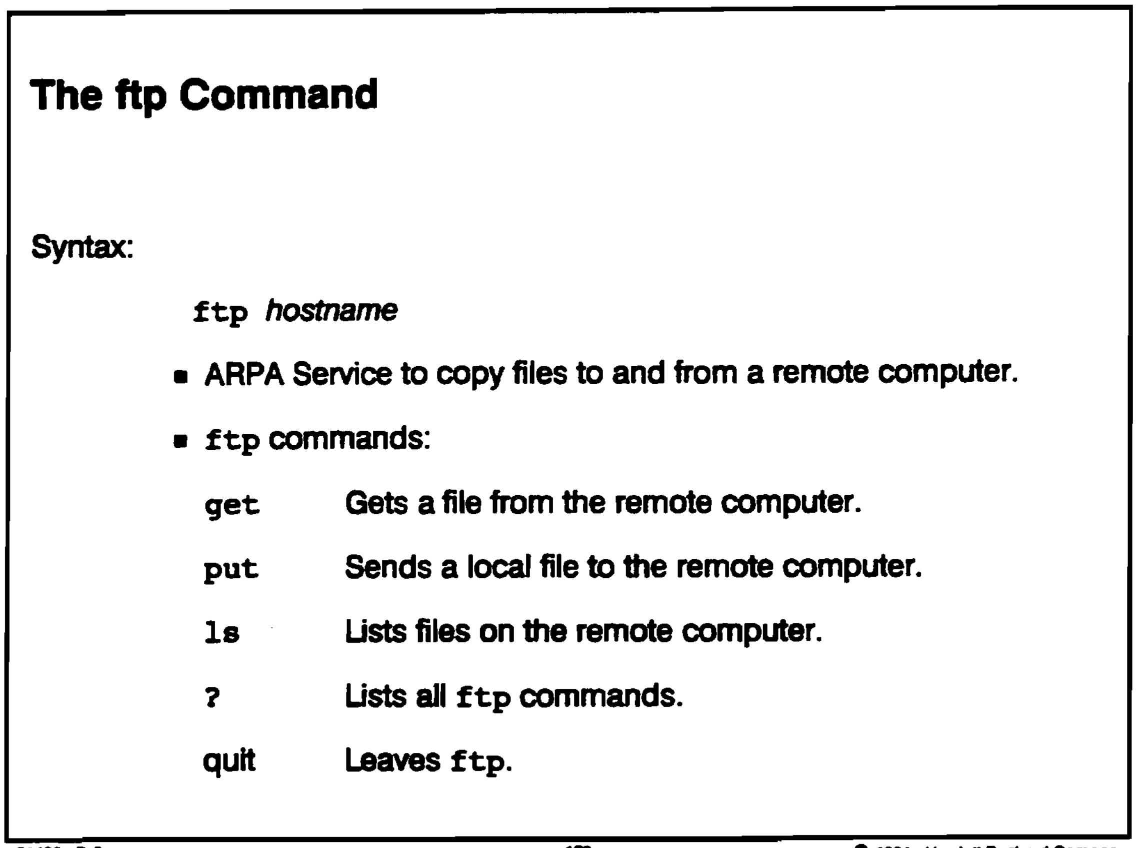
Key Points

`telnet` has many more facilities than we will discuss here. `telnet` *hostname* is the most basic form of the command.

Transition

The ARPA Service to perform remote file transfers is called `ftp`.

D-5. SLIDE: The ftp Command



The ftp Command

Syntax:

```
ftp hostname
```

- ARPA Service to copy files to and from a remote computer.
- `ftp` commands:

`get`	Gets a file from the remote computer.
`put`	Sends a local file to the remote computer.
`ls`	Lists files on the remote computer.
`?`	Lists all `ftp` commands.
quit	Leaves `ftp`.

51489 D-5. 120 © 1991 Hewlett-Packard Company

Student Notes

To copy a file to or from a remote computer using the ARPA Services, you would use the `ftp` command. `ftp` stands for "file transfer protocol." As with `telnet`, you must specify the host name of the remote machine:

```
$ ftp hostname
```

`ftp` will prompt you for your user name and password on the remote system. It *requires* that you have a password set on the remote computer. Once you give it the correct login information, you will be connected to *hostname*.

At this point you get the `ftp>` prompt. At this prompt you can use the numerous `ftp` commands to do your work. Here are a few of the common `ftp` commands for performing remote file transfers:

`get` *rfile lfile*	This copies the file *rfile* on the remote computer to the file *lfile* on your local computer. You can also use full path names as file names.
`put` *lfile rfile*	This will copy the local file *lfile* to the remote file named *rfile*.

`ls`	List the files on the remote computer. This works just like the `ls` command we have been using.
`?`	List all of the `ftp` commands.
`help` *command*	Display a brief (very brief) help message for *command*.
`quit`	Disconnect from the remote computer and leave `ftp`.

For example, you want to copy your local file called `new.jersey` to the `/tmp` directory on another computer whose host name is *fred*. Your session would look something like this (the underlined text is what you type):

```
$ ftp fred
Connected to fred.
220 fred FTP server (Version $Revision: 1.21 $DATE: 88/12/21 10:19:25 $) ready.

Name (fred:gerry): Return
Password (fred:gerry): Enter your password and press Return
331 Password required for gerry.
230 User gerry logged in.

ftp> put new.jersey /tmp/new.jersey
200 PORT command okay.
150 Opening data connection for /tmp/new.jersey (192.1.4.1,1041).
226 Transfer complete.
3967 bytes sent in 0.19 seconds (20.57 Kbytes/sec)

ftp> ls /tmp
200 PORT command okay.
150 Opening data connection for /bin/ls /tmp (192.1.4.1,1042) (0 bytes).
exercises
new.jersey
reconfig.log
update.log
226 Transfer complete.
56 bytes received in 0.36 seconds (0.24 Kbytes/sec)

ftp> quit
221 Goodbye.
$
```

The first thing you will notice about `ftp` is that it is very verbose. It has a response for every command you type. (You can tell that it was not originally a UNIX system facility!)

D-5. SLIDE: The ftp Command

Instructor Notes

Purpose

If you use the ARPA Services, you will use `ftp` to transfer files.

Key Points

`ftp` requires that the user have a password set on the remote computer.

There are many `ftp` commands to facilitate remote file transfers. ? will list the commands.

Where Problems Arise

`ftp`'s output can be very confusing to the first-time user. The example shows a typical `ftp` session.

Presentation Suggestions

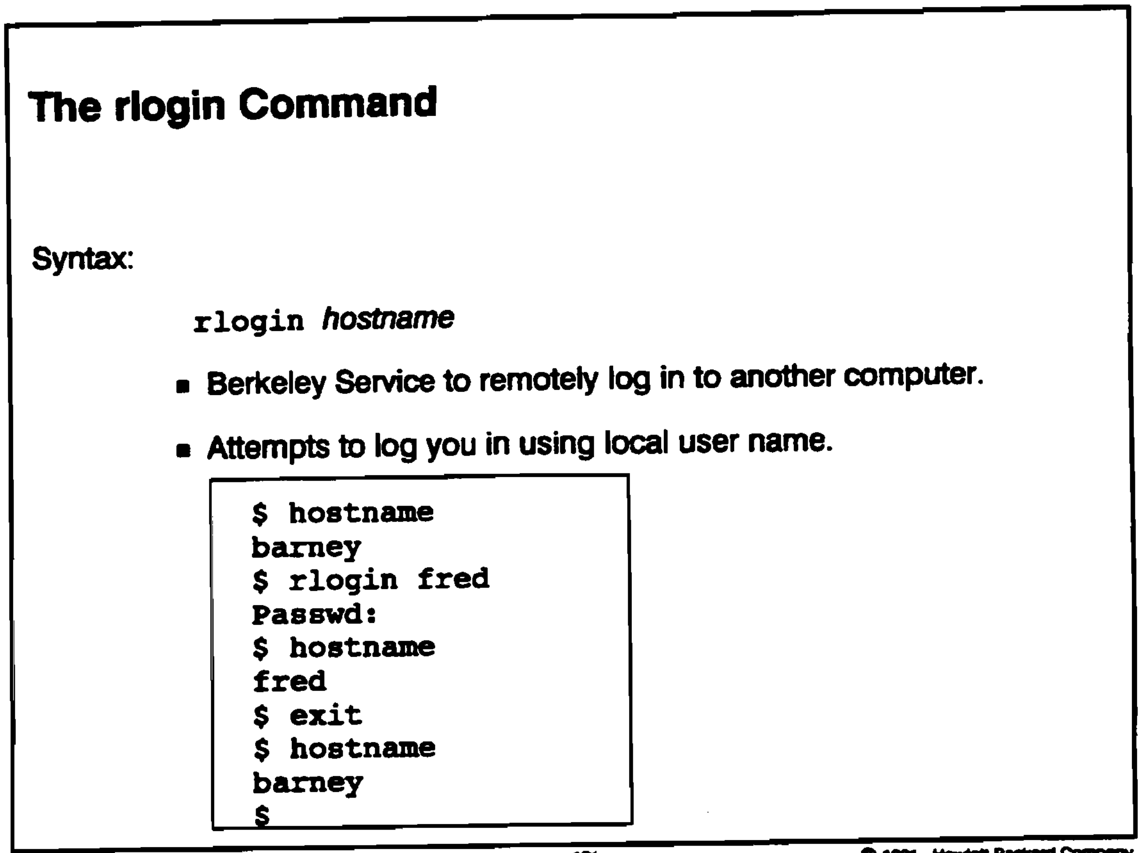
Go through the example explaining that the messages are not error messages. They are `ftp`'s way of telling you what it is doing.

You may want to explain a `get` example also.

Transition

The Berkeley Services have commands to perform similar tasks.

D-6. SLIDE: The rlogin Command



The rlogin Command

Syntax:

`rlogin` *hostname*

- Berkeley Service to remotely log in to another computer.
- Attempts to log you in using local user name.

```
$ hostname
barney
$ rlogin fred
Passwd:
$ hostname
fred
$ exit
$ hostname
barney
$
```

51489 D-6. 121 © 1991 Hewlett-Packard Company

Student Notes

The `rlogin` command performs functions similar to the `telnet` command. If you type:

```
$ rlogin hostname
```

you will be automatically logged in to the system named *hostname*. `rlogin` assumes that you are logging in to the remote computer with the same name you used to log in to the local system. As a result, it does not have to prompt you for your user name.

If your system administrator has a file called `/etc/hosts.equiv` configured, `rlogin` will not even prompt you for a password. This makes it very quick and easy to use. A file called `.rhosts` can be created in your home directory which would also let you remotely log in to that computer without using a password. See hosts.equiv(4) for more information on the format of `.rhosts`.

As with `telnet`, to disconnect from the remote computer, simply log off.

D-6. SLIDE: The rlogin Command

Instructor Notes

Purpose

`rlogin` is usually the preferred way to remotely log in to another UNIX system computer because it is easier to manipulate than `telnet`.

Key Points

The file `/etc/hosts.equiv` is a list of host names that have the same users. If your login name is "gerry" on one system, it is assumed that your are the same "gerry" on all hosts listed in `hosts.equiv`. This is helpful when `rlogin` is used, and it *must* be configured correctly when using `remsh` and `rcp`. You may want to mention `$HOME/.rhosts` as an alternative to `hosts.equiv`.

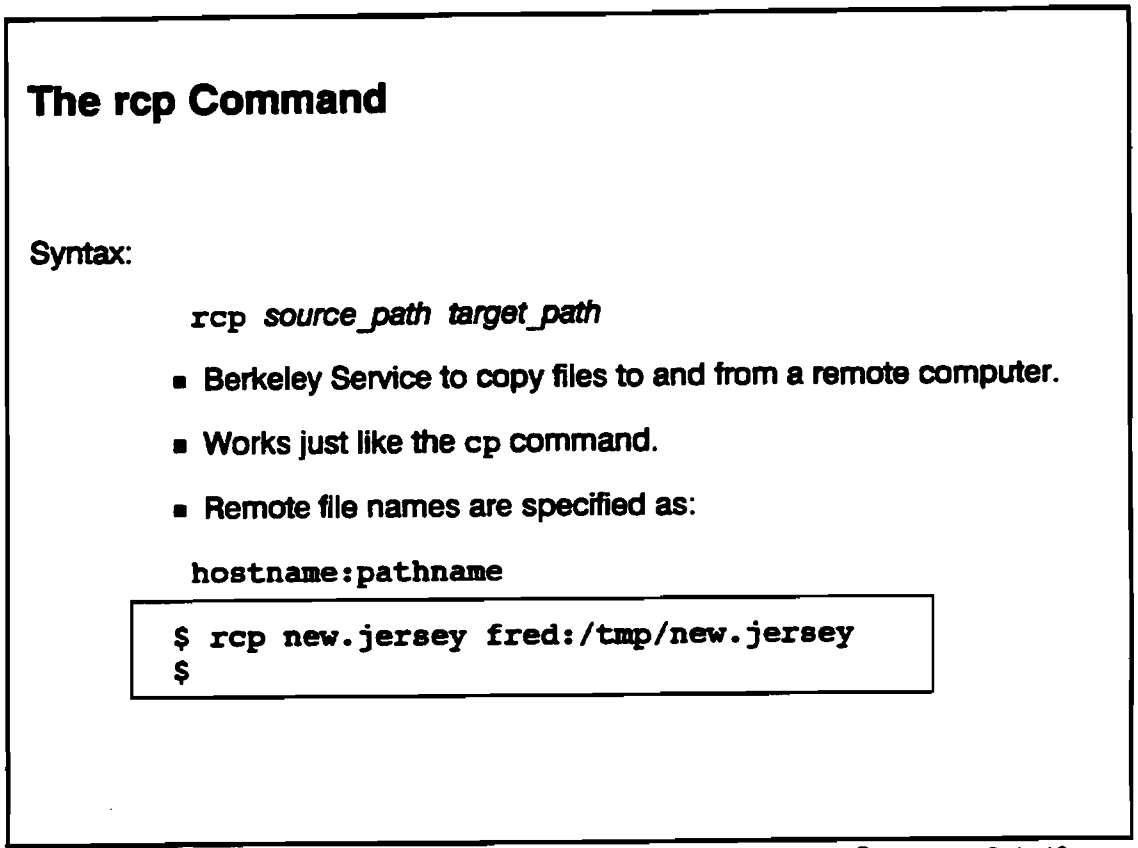
Evaluation Question

Why would you use `rlogin` instead of `telnet` and vice versa? It depends a great deal on what services you and the remote computer have in common.

Transition

Let's take a look at how to do remote file transfers using the Berkeley Services.

D-7. SLIDE: The rcp Command



The rcp Command

Syntax:

rcp *source_path target_path*

- Berkeley Service to copy files to and from a remote computer.
- Works just like the cp command.
- Remote file names are specified as:

hostname:pathname

```
$ rcp new.jersey fred:/tmp/new.jersey
$
```

51489 D-7. 122 © 1991 Hewlett-Packard Company

Student Notes

rcp stands for "remote cp." That is because it works just like the cp command. It works between two computers running the Berkeley Services. The general format of the command is:

```
$ rcp host1:source host2:dest
```

where the arguments mean copy the file *source* from *host1* to the file called *dest* on *host2*. *source* and *dest* could be full path names, of course.

If you are copying to or from a local file, you can leave off the local host name and the ":". Some examples will help make rcp more clear:

Copy the file new.jersey on the local machine (called "bambam") to /tmp/new.jersey on the system called "fred":

```
$ rcp new.jersey fred:/tmp/new.jersey
```

Copy `/tmp/new.jersey` on fred to the `/tmp` directory on barney:

```
$ rcp fred:/tmp/new.jersey barney:/tmp
```

All of the rules that apply to the `cp` command also apply to the `rcp` command.

Note The file `/etc/hosts.equiv` or `.rhosts` must be configured correctly for `rcp` to work.

D-7. SLIDE: The rcp Command

Instructor Notes

Purpose

`rcp` is a much simpler way to copy files between UNIX systems.

Key Points

Each system (bambam, fred, and barney) should have a `hosts.equiv` file that contains all three names.

You can leave off `hostname:` if the file is on the local machine.

`rcp` is easier to use than `ftp`; however, `rcp` is much slower if you are transferring a large number of files between two machines.

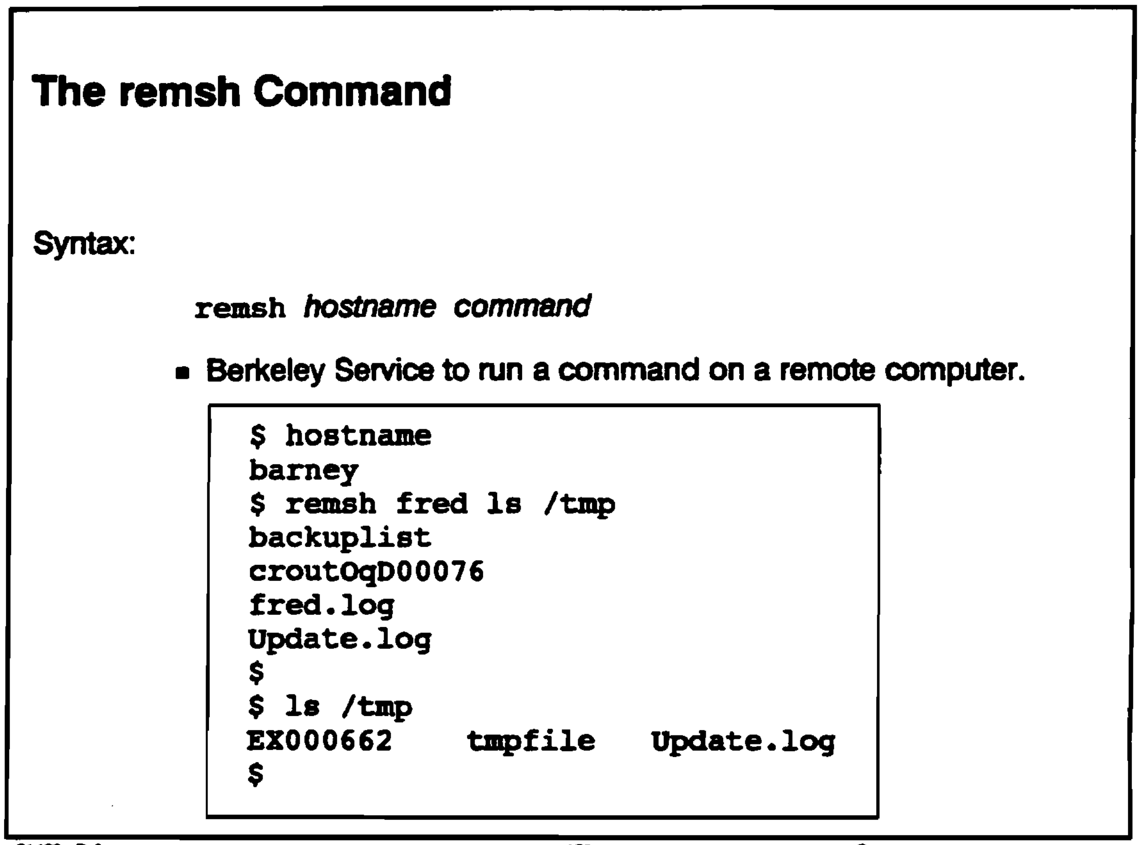
Presentation Suggestions

You could show the students a few more examples if necessary.

Transition

The last Berkeley Service that we will look at is the `remsh` command.

D-8. SLIDE: The remsh Command



The remsh Command

Syntax:

`remsh` *hostname command*

- Berkeley Service to run a command on a remote computer.

```
$ hostname
barney
$ remsh fred ls /tmp
backuplist
croutOqD00076
fred.log
Update.log
$
$ ls /tmp
EX000662    tmpfile   Update.log
$
```

51489 D-8. 123 © 1991 Hewlett-Packard Company

Student Notes

`remsh` allows you to run a program on a remote computer and see the results on your terminal. The general form of the command looks like the following:

```
$ remsh hostname command
```

For example, if you wanted to see what is running on the system "fred," you could execute:

```
$ remsh fred ps -ef
```

List the files in fred's `/tmp` directory:

```
$ remsh fred ls /tmp
fredfile
new.jersey
reconfig.log
update.log
```

Or, if you wanted to view the /etc/hosts file on fred:

```
$ remsh fred cat /etc/hosts | more
```

Notice that cat /etc/hosts is the only command being executed on fred. The output is coming to our terminal and that output is being piped to more.

You can also use remsh to print files on a printer connected to another computer:

```
$ cat myfile | remsh fred lp
```

Note The file /etc/hosts.equiv or .rhosts must be configured correctly for remsh to work.

D-8. SLIDE: The remsh Command

Instructor Notes

Purpose

`remsh` can be used in many ways to facilitate two machines working together across the LAN. It is an extremely powerful tool when used in shell programs.

Key Points

You may want to quote the command to be executed remotely to prevent the local shell from interpreting any special characters.

Transition

Let's try a few exercises.

D-9. LAB: Exercises

Directions:

Ask your instructor which exercises you can do in the classroom. Also find out the host names of the computers you can communicate with.

1. Use the hostname command to determine the name of your local system. What systems can you communicate with?

2. Use `telnet` to log in to another computer. Use the `hostname` command to verify that you are connected to the correct computer. Log off the remote computer when you are done.

3. Transfer one of your files to your home directory on a remote computer using `ftp`, and then use `rcp` to copy another file to the remote machine. Notice the differences.

4. Use `remsh` to list the contents of the remote directory to verify that the copy worked.

D-9. LAB: Exercises

Instructor Notes

Purpose

To practice using the basic LAN services.

Lab Setup

If you are in a networked classroom, make sure the students can log in to another system. Make sure the `/etc/hosts.equiv` file contains the names of all of your computers on all systems.

If you are not in a network, the students can still do the exercises on the local system. Just be careful of what file names get used in the lab.

Solutions

Directions:

Ask your instructor which exercises you can do in the classroom. Also find out the host names of the computers you can communicate with.

1. Use the hostname command to determine the name of your local system. What systems can you communicate with?

Answer:

The `hostname` command reports the local host name. By looking at the `/etc/hosts` file, you can see all of the computers your local computer can talk to.

2. Use `telnet` to log in to another computer. Use the `hostname` command to verify that you are connected to the correct computer. Log off the remote computer when you are done.

Answer:

```
$ telnet fred
Trying...
Connected to fred
Escape character is '^]'.

HP-UX fred 8.00 B 9000/350

login:  enter your name
Password: enter your password
      .
      .
      .
```

```
$ hostname
fred
$ exit
```

3. Transfer one of your files to your home directory on a remote computer using `ftp`, and then use `rcp` to copy another file to the remote machine. Notice the differences.

Answer:

In `ftp`, you would use the `put` command, similar to the example given in the student notes.

4. Use `remsh` to list the contents of the remote directory to verify that the copy worked.

Answer:

```
$ remsh system ls
```

The `ls` command will list your *home* directory on *system*.

Appendix E — Command Origin and Conformance

The following table gives the origin of each command covered in the course and the standard specifications to which that command conforms.

Command	Origin	Standards Conformance
`banner`	AT&T	SVID2, XPG2, XPG3
`cancel`	AT&T	SVID2
`cat`	AT&T	SVID2, XPG2, XPG3, proposed POSIX.2 FIPS
`cd`	AT&T	SVID2, XPG2, XPG3, proposed POSIX.2 FIPS
`chgrp`	AT&T	SVID2, XPG2, XPG3, proposed POSIX.2FIPS
`chmod`	AT&T, HP	SVID2, XPG2, XPG3, proposed POSIX.2 FIPS
`chown`	AT&T	SVID2, XPG2, XPG3, proposed POSIX.2 FIPS
`cp`	AT&T, Berkeley, HP	SVID2, XPG2, XPG3, proposed POSIX.2 FIPS
`date`	AT&T, HP	SVID2, XPG2, XPG3, proposed POSIX.2 FIPS
`echo`	AT&T	SVID2, XPG2, XPG3, proposed POSIX.2 FIPS
`env`	AT&T	SVID2, XPG2, XPG3, proposed POSIX.2 FIPS
`export`	AT&T	
`find`	AT&T, HP	SVID2, XPG2, XPG3, proposed POSIX.2 FIPS
`ftp`	Berkeley	
`hostname`	Berkeley	
`keysh`	AT&T, HP	
`kill`	AT&T	SVID2, XPG2, XPG3, proposed POSIX.2 FIPS
`lp`	AT&T	SVID2, XPG2, XPG3, proposed POSIX.2 FIPS
`lpstat`	AT&T	SVID2, XPG2, XPG3
`ls`	AT&T, Berkeley, HP	SVID2, XPG2, XPG3, proposed POSIX.2 FIPS
`mailx`	AT&T	SVID2, XPG2, XPG3
`man`	Berkeley	
`mkdir`	AT&T	SVID2, XPG2, XPG3, proposed POSIX.2 FIPS
`more`	Berkeley	
`mv`	AT&T, Berkeley, HP	SVID2, XPG2, XPG3, proposed POSIX.2 FIPS
`news`	AT&T	SVID2, XPG2

Command	origin	Standards Conformance, Cont'd.
`passwd`	AT&T	SVID2, XPG2
`pr`	AT&T	SVID2, XPG2, XPG3, proposed POSIX.2 FIPS
`ps`	AT&T, HP	SVID2, XPG2, XPG3
`pwd`	AT&T, HP	SVID2, XPG2, XPG3,proposed POSIX.2FIPS
`rcp`	Berkeley	
`remsh`	Berkeley	
`rlogin`	Berkeley	
`rm`	AT&T	SVID2, XPG2, XPG3, proposed POSIX.2 FIPS
`rmdir`	AT&T	SVID2, XPG2, XPG3
`shutdown`	AT&T	
`telnet`	Berkeley	
`vi`	Berkeley	SVID2, XPG2, XPG3
`wc`	AT&T	SVID2, XPG2, XPG3, proposed POSIX.2 FIPS
`who`	AT&T, HP	SVID2, XPG2, XPG3

Appendix F — Lab Notes For MPE Users

This appendix provides additional notes and explanations for students who have worked with the HP3000. The goal of this appendix is to relate HP-UX to the MPE user's past experience by highlighting similarities and differences between HP-UX and MPE. This should allow MPE users to draw on their prior knowledge to develop HP-UX skills more quickly and competently. This should also reduce areas of confusion.

The appendix is broken up into several sections:

- Logging In
- Commands
- File System
- Permissions
- Text Editing
- Shell Programming
- Batch Processing

These notes are designed as a self study to be read independently by the student either before or after appropriate labs.

Logging In

On MPE, a login id consists of two required pieces of information; an MPE user name and an MPE account name. In addition, a third identifier, the MPE group name, can optionally be used to specify which MPE group the user logs into. The group name is required if the user has no home group.

On HP-UX, a login id consists of only a username.

In MPE, the user, account, and group name information is stored in the MPE System Directory. This information is accessible through LISTDIR5 on MPE V or, on MPE XL, :LISTUSER, :LISTACCT, :LISTGROUP; and is maintained through :NEWUSER, :ALTUSER, :PURGEUSER, :NEWACCT, :ALTACCT, :PURGEACCT, :NEWGROUP, :ALTGROUP, :PURGEGROUP.

In HP-UX, all information related to a user name is kept in the file /etc/passwd and can be maintained by editing this ASCII file. HP-UX also provides sam, a menu driven system administrator utility which edits this file for you. The user names and associated information are accessible simply by looking at the contents of this file. That is:

```
cat /etc/passwd
```

Section four of the HP-UX reference manual documents the layout of /etc/passwd. Each line contains a separate username. Several fields, separated by colons, are stored as follows:

```
sally:tekXeRorZ0qXQ:201:20::/users/planets/sally:/bin/ksh
```

- login name (`sally` from example above)

  User name. All user names as well as file names and commands are case sensitive in HP-UX.

- encrypted password (`tekXeRorZ0qXQ` from example above)

  In MPE, a user could be prompted for as many as three different passwords at login time, for the account password, user password, and group password. At most one password is required for HP-UX. In HP-UX, all passwords are stored in an encrypted format, so no one, not even super-user, can read these passwords. The super-user can, however, edit `/etc/passwd` to remove an encrypted password. On HP-UX, users maintain their own passwords through the `passwd` command.

- numerical user id (201 from example above)

  Each user name has associated with it a userid number. A 0 signifies super-user privilege. Super-user privilege can be assigned to one or more HP-UX users. Super-user privilege allows access to special administrative tools and overrides many operating system controls. MPE capabilities can be assigned selectively to MPE users. The MPE user `MANAGER.SYS` has full MPE capabilities. So too, the HP-UX user `root` has super-user privilege. Having super-user privilege on HP-UX is similar to having a combination of SM, PM, and OP capabilities on MPE.

- group id number (20 from example above)

  This is the id number of the group this user initially belongs to. Both MPE and HP-UX use the term group. In MPE a group is a collection of files. We will see that on HP-UX a collection of files is called a directory.

  In HP-UX a group is a collection of users. Groups are used for security purposes. Each file is owned by a group. A group, that is all its members, can be assigned permission to access the files the group owns. And access can be denied to other users of the system who are not members of the group.

  Each user is permitted to be a member of one or more groups (one group at a time).

  Let's say we have a group named `payroll` with the user names `sally`, `hank`, and `sue` permitted to be members. The file `/employee/pay` is owned by the `payroll` group. I allow only members of the `payroll` group read access to `/employee/pay`.

  The user `sally` may also be permitted to be a member of a group named `benefits`. Only members of the `benefits` group are permitted read access to the file `/employee/ben`. `/employee/ben` is owned by the `benefits` group.

  Let us say `sally` is initially assigned to the `payroll` group. That is, the fourth field in `/etc/passwd` for `sally` has the `payroll` group's id number.

  So when sally issues the command `id` she would see her current user id is `sally` and her current group id is `payroll`. At this point, `sally` is permitted access to the file `/employee/pay` but is denied access to the file `/employee/ben`. To switch to the `benefits` group and gain access to the file `/empoyee/ben` `sally` would issue the command:

  ```
  newgrp benefits
  ```

  Now the `id` command will reflect that her user id is still `sally` but her group id is now `benefits`; thus now allowing her access to the file called `/employee/ben`.

The list of user-group relations is contained in the file, `/etc/group`. Specifically, each line contains a group name, an optional password, a group id number, and a list of user names permitted as members of the group. This information is accessible by looking at the contents of this ASCII file and is maintained by editing it with any text editor such as `vi`. The section of the file we described would look like:

```
payroll::20:sally,hank,sue
benefits::21:sally
```

- reserved field (empty in example above)

  May be used for personal identification such as user's full name, office location, extension, home phone.

- initial working directory (`/users/planets/sally` in example above)

  This is similar in concept to a home group in MPE. When a user logs on, he is initially placed in this home directory. Relative pathnames will be evaluated relative to this directory. On MPE XL, the `:CHGROUP` could be used to switch to other groups of files without logging off. On HP-UX, a user can issue a `cd` (change directory) command to switch directories. We will see that the user does need appropriate permissions (file security) to a directory in order to switch to it.

- shell program (`/bin/ksh` in example above)

  This is the shell program or command interpreter that will be run for this user upon login. A command interpreter is the part of the operating system that reads the user's commands and initiates execution of these commands. MPE provides only one command interpreter, but on HP-UX there are several command interpreters (or shells) to choose from. The primary shells available are:

  — Bourne shell (`/bin/sh`) oldest, default shell

  — C shell (`/bin/csh`) Berkeley shell

  — Korn shell or K shell (`/bin/ksh`) newer shell, incorporates best of C shell extensions and maintains compatibility with Bourne shell.

The HP3000 system manager must specify in the configuration the type of terminal each user is working with. That means MPE knows what type of terminal you are using before you login.

HP-UX, on the other hand, determines what type of terminal you are using at the time you login. UNIX was originally designed to run on virtually any hardware, using a variety of terminals. Some older terminals supported on HP-UX don't have backspace keys and have only uppercase. For these reasons terminals work in a special way at login time on HP-UX. In order to allow users to correct errors during login in a consistent manner regardless of what terminal they are working with, the backspace key is not used. Instead of backspace, a single character is erased with # and the entire line is erased with @. Also, if uppercase is entered at login time, HP-UX assumes you are using a terminal that is strictly uppercase and all characters are shifted to lower case before being executed. To avoid this, user names should always be entered in lower case when logging in.

Commands

Many HP-UX commands have MPE commands that are similar.

HP-UX	MPE
touch	build
cat	fcopy (to screen)
more	print (XL)
ls	listf
cp	fcopy, copy (XL)
mv	rename
rm	purge
mkdir	newgroup
rmdir	purgegroup
who	showjob job=@s (sessions only)
whoami	showme
ps	showjob, showproc (XL)
mail	HPDESKMANAGER (the product)
pwd	showme (to see the group name)
id	showme (to see the user name)
write	tell
cd	chgroup (XL)
chmod (on a file)	altsec
chmod (on a directory)	altgroup access=, altacct access=
ln	no equivalent
chown	no equivalent
vi	editor
sort	sort
nice	job ;pri=es
kill	abortjob
if	if
tar	store/restore
man	help
fc -l (Korn shell)	listredo (XL)
r (Korn shell)	do (XL, redo without editing)
read	input (XL)
nohup (with & as suffix)	stream
at (no & required)	stream ;at=

The `man` command is similar to MPE `:HELP`. It allows on-line access to the HP-UX reference manual.

Many MPE users are familiar with UDCs (or command files) that are abbreviations for commands. Many MPE users create UDCs for longer command lines in an effort to save keystrokes. For example:

```
SJ = SHOWJOB
SJJ = SHOWJOB JOB=@J
SJS = SHOWJOB JOB=@S
```

HP-UX was designed with similar time savers already built in, ie, cp = copy, mv = move, cd = change directory, pwd = present working directory, man = manual, etc. On HP-UX you must use these abbreviations.

Many UNIX commands can operate in a slightly different way by providing options. The man command contains a -k option which allows for an on-line reference manual search based on a key word.

The MPE `:LISTREDO` and `:REDO` commands provide access to commands the MPE user has previously executed. In the HP-UX Korn shell, (`/bin/ksh`), the escape key followed by the k key brings up the last command the HP-UX user just executed. At this point the user can use `vi` commands to edit the line displayed before re-executing this line. By entering the k key over and over the user can retrieve earlier commands that had been executed and can edit them with `vi`.

File System

On MPE, all files reside within a MPE group. Each group resides within a MPE account. So all files can be identified by "filename.groupname.accountname". The lists of valid accounts, groups and files are all located in the MPE system directory.

On HP-UX, the term "directory" is used to mean something quite different from the MPE system directory. A directory on HP-UX is simply a collection of files; very much like a MPE group is a collection of files. There are, however, some differences. MPE groups contain only files. I cannot have one MPE group contained within second MPE group. On HP-UX, however, I can have one directory stored within another directory. That is, a directory can contain files and/or directories. Looking back, MPE could be thought of as a two level directory file system where the "upper level" directory is the account name and the "lower level" directory (or sub-directory) is the group name.

HP-UX has a parent directory for all files and directories on the system called root, written as /. So all files and directories are contained within /.

On MPE a fully qualified filename is "filename.groupname.accountname".

On HP-UX the absolute path name is similar to this fully qualified filename. All directories that the file is located within are specified. On MPE, the filename is listed first, followed by the sub-directory (group), followed by the parent directory (account). On HP-UX the sequence is reversed. The absolute path name always starts with the highest level directory containing the file which will always be /. The / is followed by all other directories containing the file with the higher level directories preceding the lower level directories. All intermediary directory names are suffixed with a /. The last piece of information is the file name.

So `/etc/passwd` refers to the absolute pathname for the file `passwd` which is located within the directory `etc`. `etc` is located within the directory / (root).

The file `/users/planets/earth/new.jersey` refers to the file `new.jersey` (notice the HP-UX file names allow special characters like a . and can be longer than eight characters) which is located within the directory `earth`. Remember HP-UX is case sensitive so `Earth` is not the same directory as `earth`. `earth` is located within the directory `planets`; `planets` is located within the directory `users`; `users` is located within the directory /.

On MPE, if the file we want to access resides within our logon group and account, we do not need to fully qualify the file name to access it. So if we were logged into the group `DB` within the account `MFG` and we wanted to access the file `TRS` (full file name is `TRS.DB.MFG`) we could specify simply `TRS`.

Similarly, on HP-UX, we can specify a relative pathname as opposed to an absolute path name. We do this by specifying only the portion of the pathname that is "below" our present working directory.

So if our present working directory were `/users/planets/earth` and we wanted to view the contents of the file `/users/planets/earth/new.jersey` with the cat command, we could simply enter:

```
cat new.jersey
```

as opposed to writing out

```
cat /users/planets/earth/new.jersey
```

which would also work.

PERMISSIONS

On MPE, file security includes provisions to differentiate between five types of file access: read, write, lock, append and execute. Also, the ability to create new files is controlled by assigning save access to a group.

Additionally, on MPE, in order to be able to access the file the user must have the appropriate access at the account, group and file level.

Finally, on MPE, users are segregated into categories, Account Member, Group User, Any, Account Librarian, Group Librarian and Creator.

On HP-UX, read, write, and execute permissions may be assigned to both files and directories. HP-UX segregates users into three mutually exclusive categories:

- user (owner of the file or directory). Every file and directory has an owner. Ownership can be changed with the chown command.
- group (every file and directory is owned by a group). All logons who are currently members of the group that owns the file fall in this category. Group ownership of a file can be changed with the chgrp command.
- other (anyone else on the system).

The HP-UX user is similar to MPE creator. HP-UX group membership is initially based on the group id entry in the `/etc/passwd` file and can be changed with the newgrp command.

Read, write and execute mean something special when applied to directories.

Read access to a directory means that we can use the ls command to get a list of files in that directory. On MPE the LISTF command is always available to get a list of files from any group/account and cannot be disabled through file security.

Write access to a directory means that we can add or remove files from the directory. On MPE, save access to a group allows us to add files to the group. On MPE, the ability to purge files is strictly limited to creator and SM capability and cannot be further controlled with file security.

On MPE XL we need to know the group password to issue the CHGROUP command to switch to a different group. In order to use the HP-UX cd command to switch to another directory we need execute permission to the directory we are switching to.

Similar to MPE, having read access to a file at the file level is not sufficient access for us to read the file. We also need appropriate access to the directory in which the file resides and to all additional directories included in the pathname of the file.

So, in order to access a file in any way (read,write or execute) not only do we need appropriate access to the file; we also need search access (execute access) to that file's parent directory and to any additional directories included in that file's pathname.

For example, let's say our present working directory is `/users` and we are trying to read the file `/users/planets/earth/new.jersey`. In order for our command

```
cat planets/earth/new.jersey
```

to succeed, we need read access to the file `new.jersey` and search access (that is, execute access) to the directories `planets` and `earth`.

Text Editing

Many editors are available on both MPE and UNIX. EDITOR is available on every MPE system and for that reason is taught in introductory MPE classes. `vi` is taught in this class because it is so widely available on UNIX systems.

MPE EDITOR is a line editor. `vi` works differently. `vi` relies heavily on command mode and input mode. In EDITOR the add command allows lines to be added to the text. Everything that is typed after entering the add command is included as contents of the file until you key //. In `vi` this is called input mode. In EDITOR you terminate "input mode" for the add command by typing //. In `vi` several commands put you into input mode. For example, a, i, o, and O all initiate input mode. In each case, everything typed from that point until the escape key is hit is added to the file. So the escape key in `vi` serves a similar purpose as the // serves for the add command in EDITOR.

Shell Programming

A MPE UDC catalog is a file that can contain multiple command definitions. For example:

```
purgeudc
file a=listout
purge 1
run purgprog
****
buildfil
    .
    .
    .
****
anotherudc
    .
    .
    .
****
```

UNIX provides the ability to create our own commands through shell programs. Both the body of a UDC and the contents of a shell program are made up of operating system commands. One MPE UDC file can contain multiple command definitions as in the example above. A HP-UX shell program contains the definition of a single command. The command is executed by typing the name of the HP-UX file. Shell programs can use parameters just as UDCs can. The following UDC and shell program both purge four files.

```
purgeudc f1,f2,f3,f4
purge !f1
purge !f2
purge !f3
purge !f4
*

purgeshellprogram
rm $1
rm $2
rm $3
rm $4
```

In MPE several commands including `:LISTF`, `:STORE`, `:RESTORE` allow for wildcards in filenames where @ matches any pattern, ? matches any single character, and # matches any digit. HP-UX has a similar file name matching facility which works on all commands where * matches any pattern (* = @), ? matches any single character (? = ?), and [ ] can be used to identify a character class to match a single position. So [0-9] matches any single digit.

MPE	HP-UX	
`?###`	`?[0-9][0-9][0-9]`	*any character followed by three digits*
`@a@`	`*a*`	*a anywhere in filename*
`@t`	`*t`	*ends with t*

Many features of UNIX such as redirection of command input and output, command variables, command files (shell programs), implied run, the PATH variable (called HPPATH), and character classes are part of MPE XL.

The logon option of MPE UDCs allows a UDC to be executed automatically when the user logs into MPE. The `.profile` file in the user's home directory is executed automatically when a HP-UX user first logs in and provides similar functionality.

The nobreak option of a MPE UDC disables the break key. The HP-UX `trap` command can be used to provide the same functionality in a shell program.

Batch Processing

The :STREAM command on MPE can be used to initiate a batch job. The batch job logs on independent of the on-line user who issued the STREAM command. A HP-UX command that has a & at the end will run as a background process. This background process is similar to a batch job in that both allow the user to continue working interactively and both share the CPU with any on-line users.

The HP-UX background process is not independent of the on-line user. The MPE batch job sends all its default output ($STDLIST) to the printer. The HP-UX background process sends all its default output (stdout) to the user's screen. If the MPE user who streamed the job logs off, the job continues to execute. If the HP-UX user who initiated the background process logs off, the background process will abort.

The HP-UX nohup command allows the HP-UX background process to work more like an MPE job in that nohup processes will continue to execute after the on-line user logs off.

For example, in MPE

```
STREAM MYJOB
```

initiates a batch job.

In HP-UX

```
nohup myjob &
```

initiates the background process.

The HP-UX at command provides the ability to schedule jobs to run at a future time similar to the MPE command STREAM ;AT=. The background process will execute at the future time even if the user logs out.

In order to remove a job or session on MPE we would first issue a :SHOWJOB to determine the job or session number and then issue an :ABORTJOB to abort the job or session. The HP-UX user would first issue a ps command to determine the process id of the background process or on-line user and then issue a kill command to send the process a signal causing the process to abort. Signal number 9 is a "sure kill" and cannot be disabled by the trap command. For example:

on MPE:

`SHOWJOB`	*determine which job or session number to abort*
`ABORTJOB #J154`	*remove job number #J154 from the system*

on HP-UX:

`ps -ef`	*determine which process id to abort*
`kill -9 16882`	*remove process id number 16882 by sending it signal 9*

Solutions

2-5. LAB: Exercises

This is an exercise in HP-UX hardware identification. For the system being used in this class, identify the following items:

1. The computer model that is being used.

Answer:

The computer model used is:

2. The device that is being used as a system console.

Answer:

The device is:

3. The disk or disks that are connected to the system.

Answer:

The disk or disks connected to the system are:

4. The tape device that is being used.

Answer:

The tape device being used is:

5. The maximum number of terminals that can be connected to the system.

Answer:

The maximum number of terminals is:

3-6. SLIDE: The date Command

1. You should still be logged in. Try to report the date and time by executing the `date` command at the shell prompt.

Answer:

```
$ date
```

3-7. SLIDE: The who Command

1. Try each of the following commands and note the difference in output: `who`, `who am i`.

Answer:

You should see output similar to the examples on the slide:

```
$ who
gerry     console      May 25 10:06
kevin     tty1p0       May 25 13:10
ginger    tty1p1       May 25 08:04
renate    tty1p2       May 24 08:29
kevin     pty/ttyp2    May 25 10:10
$ who am i
kevin     pty/ttyp2    May 25 10:10
```

3-12. SLIDE: Reference Manual

1. If you have printed copies of the *HP-UX Reference Manual*, try to find the `who` command.

Answer:

Since it is a user command, you would look under who(1).

3-14. SLIDE: The man Command

1. How would you find more information on `man` using the `man` command?

Answer:

```
$ man man
```

2. Try to get information on the `date` command using `man`.

Answer:

```
$ man date
```

3-15. LAB: Exercises

Each person should do these exercises once. If you have a partner, each of you should do each exercise at the terminal.

1. Practice logging off. Then log back in again.

Answer:

$ Ctrl-d Return or $ `exit` Return

2. Run the `hello` program that is in your home directory.

Answer:

```
$ hello
```

3. Look in the reference manual to find out which `who` option is used to print column headings above the output. Try `who` using this option.

Answer:

```
$ who -H
NAME       LINE         TIME
gerry      tty0p2       May 25 10:06
jimd       tty1p0       May 25 13:10
doug       tty1p1       May 25 08:04
frank      tty1p2       May 24 08:29
```

4. Using the `echo` command, display `HP-UX is fun` on your screen. Try this again, making sure the words are three spaces apart.

Answer:

```
$ echo HP-UX is fun
HP-UX is fun

$ echo "HP-UX   is   fun"
HP-UX   is   fun
```

5. Using the banner command, display the words Hi There in large letters on your screen. Try this again, making sure both words are displayed on the same line.

Answer:

```
$ banner Hi There
#     #
#     #     #
#     #     #
#######     #
#     #     #
#     #     #
#     #     #

#######
   #     #    #  ######  #####   ######
   #     #    #  #       #    #  #
   #     ######  #####   #    #  #####
   #     #    #  #       #####   #
   #     #    #  #       #   #   #
   #     #    #  ######  #    #  ######

$ banner "Hi There"
#     #                 #######
#     #     #              #     #    #  ######  #####   ######
#     #     #              #     #    #  #       #    #  #
#######     #              #     ######  #####   #    #  #####
#     #     #              #     #    #  #       #####   #
#     #     #              #     #    #  #       #   #   #
#     #     #              #     #    #  ######  #    #  #######
```

6. Send a mail message to your partner. Read the mail that they sent to you. Now save that mail message to a file called mail.save. What command did you use?

Answer:

```
? s1 mail.save
```

7. Read the system's news. What command did you use?

Answer:

```
$ news
```

Advanced:

8. How can you make the date command display the date in mm/dd/yy format?

Answer:

```
$ date +%D

OR

$ date +%m/%d/%y
```

4-2. SLIDE: The ls Command

1. Try the `ls` command. You should see the files in your current directory. How does the output of the `ls -F` command differ from that of `ls`? Try both commands.

Answer:

`ls -F` displays a "/" after directory names and a "*" after program names.

2. How would you obtain a full listing of the /tmp directory?

Answer:

Provide /tmp as an argument to the `ls -F` command:

```
$ ls -F /tmp
./   ../   .X11-unix/   data   junk   reconfig.log
update.log
$
```

4-8. LAB: Exercises

1. Change your current directory to `/tmp` and execute the `pwd` command. What do you see? Return to your home directory and execute a `pwd`. What is the full path name of your home directory?

Answer:

```
$ cd /tmp
$ pwd
/tmp
$ cd
$ pwd
/users/login_name
$
```

2. Under your login directory there is a directory called `sample.tree`. It goes down several levels. Diagram the files and directories under `sample.tree` using the `cd`, `ls -F`, and `pwd` commands as necessary. Use ovals to indicate directory names and rectangles for regular files. Some exercises in the next module refer to files and directories somewhere under `sample.tree`.

Advanced: What option to `ls -F` could produce the names we needed using one command?

Answer:

This is what `sample.tree` should look like. You could also use `ls -FR`.

```
                              sample.tree
                                 (dir)
                                   |
      -------------------------------------------------------------
       |           |         |           |               |        |
   car.models   collie     honda     dog.breeds       poodle   porsche
     (dir)                              (dir)
       |                                  |
  ----------------                  ---------------
    |          |                     |         |
 chevrolet    ford               retriever shepherd
   (dir)      (dir)                (dir)     (dir)
                |                    |
         -------------           -------------------------
         |           |           |            |         |
       sedan       sports      golden     labrador   mixed
       (dir)       (dir)
                     |
                  mustang
```

3. What is the full path name of the file golden under the directory `retriever` in the diagram from the previous exercise? What is its relative pathname from your home directory?

Answer:

The full path name is: `/users/login_name/sample.tree/dog.breeds/retriever/golden`
The relative path name is: `sample.tree/dog.breeds/retriever/golden`

4. From your home directory, change into the `sample.tree/dog.breeds/retriever` directory. Using a relative path name, change into the `sample.tree/car.models/ford/sports` directory. Again using a relative path name, change into the `sample.tree/car.models/chevrolet` directory. Finally, return to your home directory. What commands did you use? How did you know if you arrived at each of your destinations?

Answer:

```
$ cd sample.tree/dog.breeds/retriever
$ cd ../../car.models/ford/sports
$ cd ../../chevrolet
$ cd
```

To verify each destination:

```
$ pwd
```

5. From your home directory, obtain a directory listing of the directory `dog.breeds` under the `sample.tree` directory. Use both relative and absolute path names. What commands did you use?

Answer:

```
$ ls sample.tree/dog.breeds
$ ls /users/login_name/sample.tree/dog.breeds
```

6. Place yourself in your home directory. With one command, obtain a long listing of the file cp in the /bin directory. (Hint: Use the ls -l (or ll) command.) Look at the slide in topic 4-3 for a diagram of where /bin and cp are located.

Now, figure out the relative path from your home directory to the /bin directory. Obtain the same long listing of cp using a relative path name.

Answer:

```
$ cd
$ ls -l /bin/cp
-r-xr-xr-x   1 bin      bin        27232 Feb 17 14:00
/bin/cp

<< Relative path from a standard home directory (/users/logname) is ../../bin >>
$ ll ../../bin/cp
-r-xr-xr-x   1 bin      bin        27232 Feb 17 14:00
../../bin/cp
$
```

5-3. SLIDE: The cat Command

1. Display the file murphy on your terminal using the cat command. What did you notice?

Answer:

The cat command simply concatenaters the file to your screen without regard to the size of the screen. It scrolls the text by until it reaches the end of the file

5-4. SLIDE: The more Command

1. You have already used the `cat` command to display `murphy` on your screen. Now try to display `murphy` with the `more` command.

Answer:

```
$ more murphy
```

Notice that the file is displayed one screen at a time. The space bar displays the next screen and (Return) displays just the next line.

2. Use the `more` command to display both `cbt` and `reminder` at once.

Answer:

```
$ more cbt reminder
:::::::::::::
cbt
:::::::::::::
Computer Based Training is an interactive instructional
experience between a computer and a learner in which the
computer provides the increased knowledge or skill results.
:::::::::::::
reminder
:::::::::::::
Your mother's birthday is November 29.
$
```

Notice that `more` prompted you to start displaying the second file after it displayed the first file.

5-10. LAB: Exercises

Use the *HP-UX Reference Manual* to look up details about commands and their usage.

1. Notice the two files named `display1` and `display2`. See what each contains by displaying their contents on your terminal. Try to accomplish this using a single command line.

Answer:

```
$ more display1 display2
::::::::::::::
display1
::::::::::::::
This is the file display1
::::::::::::::
display2
::::::::::::::
```

```
This is the file display2
```

Note that cat would have worked also without the separation between files.

2. Make a copy of display1 and call it More_Than_14_Chars. Now do an ls. What happens?

Answer:

```
$ cp display1 More_Than_14_Chars
$ ls
display1 More_Than_14_C
```

If you are on a system using long file names, the complete file name will be used.

3. Make a copy of display1 and call it display1.copy. Make a copy of display2 and call it display2.copy.

Answer:

```
$ cp display1 display1.copy
$ cp display2 display2.copy
```

4. Use the ls command to verify that both display1.copy and display2.copy exist. Do this so that the ls command outputs only the names of the files display1.copy and display2.copy and not the names of any other files.

Answer:

```
$ ls display1.copy display2.copy
display1.copy    display2.copy
```

5. Try to make a copy of display1.copy and call it display2.copy. Display the contents of display2.copy. What happened?

Answer:

```
$ cp display1.copy display2.copy
$ cat display2.copy
This is the file display1
```

The contents of display2.copy were overwritten with display1.copy.

6. Rename display2.copy as display1.copy1.

Answer:

```
$ mv display2.copy display1.copy1
```

7. Now remove all of the files that you have been responsible for creating. Use a single command line.

Answer:

```
$ rm display1.copy1 display1.copy More_Than_14_c
```

8. Change your current directory to `sample.tree`. In one command, copy `collie` and `poodle` into `dog.breeds`. Perform this from whatever directory is most convenient. You might need to look up the `cp` command. (Hint: `dog.breeds` will be the *target* of the copy.)

Answer:

```
$ cp collie poodle dog.breeds
```

9. In one command, move `honda` and `porsche` into `car.models`. Perform this from whatever directory is most convenient. You might need to look up the `mv` command.

Answer:

```
$ mv honda porsche car.models
```

10. In one `ls` command, verify that the `cp` and `mv` worked correctly.

Answer:

```
$ ls . car.models dog.breeds
.:
car.models    collie    dog.breeds    poodle

car.models:
chevrolet    ford    honda    porsche

dog.breeds:
collie      poodle      retriever      shepherd
```

11. Rename the `sample.tree` directory to `cars+dogs`.

Answer:

```
$ cd
$ mv sample.tree cars+dogs
```

12. Use the `rm` command with the `-i` option to remove the file `rm_me`.

Answer:

```
$ rm -i rm_me
rm_me?  y
$
```

Advanced Exercises

1. The ln command is used to attach more than one name to the same file space. If two files are linked, you can change one and both change. This is handy for sharing common files such as customization files. ln uses the same syntax as the cp and mv commands.

In your home directory there is a file called reminder. Link it to another name such as reminder.link. cat both files to prove that they are identical.

Change reminder with the command:

```
echo "Meeting 6/21" >> reminder
```

(We will see how this works in Module 9.) cat both files again. What do you see?

Answer:

```
$ ln reminder reminder.link
$ cat reminder
Your mother's birthday is November 29.
$ cat reminder.link
Your mother's birthday is November 29.
$ echo "Meeting 6/21" >> reminder
$ cat reminder
Your mother's birthday is November 29.
Meeting 6/21
$ cat reminder.link
Your mother's birthday is November 29.
Meeting 6/21
```

Notice that you changed only the file reminder but because the two files were linked, reminder.link was also changed.

2. What happens if you remove reminder.link?

Answer:

```
$ rm reminder.link
$ ls reminder reminder.link
reminder.link not found
reminder
$
```

Notice that the file names really are referencing the same space on the disk. Removing one of the links, however, leaves the other file intact.

6-11. LAB: Exercises

1. Create two directories under your home directory named mail and junk. Obtain a long listing of the contents of the two directories with the ls -al command. Note what files you see.

Answer:

```
$ cd
$ mkdir mail junk
$ ls -al mail junk

junk:
total 4
drwxrwxrwx   2 bob     aeo          64 Jun 16 11:01 .
drwxrwxrwx   6 root    other      1024 Jun 16 11:01 ..

mail:
total 4
drwxrwxrwx   2 bob     aeo          64 Jun 16 11:01 .
drwxrwxrwx   6 root    other      1024 Jun 16 11:01 ..
```

Notice that the mkdir command automatically created "." and ".." in each directory. The -a option to ls -l allows us to view file and directory names with a leading ".".

2. Copy display1 and display2 into the junk directory. Change the permissions of junk so that you do not have read permission. Verify that the new permissions have been set. Obtain a long listing of junk. What happens?

Advanced:

There is an option to the ls -l command that makes it easy to list just the information about a particular directory without listing its contents. Try this option on the junk directory.

Answer:

```
$ cp display1 display2 junk
$ chmod 377 junk
$ ls -l junk
junk unreadable
total 2
$ ls -ld junk
d-wxrwxrwx   2 bob     aeo          1024 Jun 21 14:50 junk
$
```

Since you do not have read permission on the directory, you cannot look at the contents of the directory.

Notice that the -d option allows us to view a directory itself instead of its contents.

chmod u-r junk would also have worked.

3. Change the permissions of `junk` so that you have read and write permission, but no execute permission. Obtain a long listing of `junk`. What happens?

Answer:

```
$ chmod 677 junk
$ ls -l junk
display1/junk not found
display2/junk not found
total 0
```

Since you do not have execute permission on the directory, you cannot search the directory.

4. Change the permissions of `junk` back to their default permissions. Change the permissions of `display1` in the `junk` directory so that you do not have write permission on the file. Remove `display1` from the `junk` directory. What happens?

Answer:

```
$ chmod 777 junk
$ cd junk
$ chmod 466 display1
$ rm display1
display1: 466  mode ? (y/n) y
```

Since you do not have read permission on the file, you are prompted to proceed with the removal.

5. Remove the junk directory.

Answer:

```
$ rm display2
$ cd ..
$ rmdir junk
$
```

Note that since `junk` was not empty, you first had to remove the files that it contained. Note that `rm -r junk` would have worked also.

6. Obtain a long listing of the files in your home directory staring with the letter "d."

Answer:

```
$ ls -l d*
-rw-rw-r--    1 bob    aeo                204 Mar 17   1988 delete
-rw-rw-r--    1 bob    aeo                 22 Mar 17   1988 display1
-rw-rw-r--    1 bob    aeo                 22 Mar 17   1988 display2
$
```

7. Obtain a listing of all files in your home directory with the letter "t" somewhere in their names.

Answer:

```
$ ls -l *t*
-rw-rw-r--   1 bob    aeo                35 Mar 17  1988 add.text
-rw-rw-r--   1 bob    aeo               172 Jan 10 16:11 cbt
-rw-rw-r--   1 bob    aeo               204 Mar 17  1988 delete
-rw-rw-r--   1 bob    aeo               127 Mar 17  1988 mutant
-rw-rw-r--   1 bob    aeo              6842 Mar 17  1988 unwanted
```

8. What is the command to change the ownership of `cbt` to your mail partner? Do it now. Can you get ownership back? How?

Answer:

```
$ chown mail_partner_login_id cbt
```

You can only get ownership back if your mail partner gives it back to you.

9. From your home directory, copy the file `mutant` to the directory `/bin`. Did you have any problems? If so, why?

Answer:

The `/bin` directory does not have *write* permission set for others.

10. Copy the file `/bin/cp` to your home directory. Did you have any problems? If so, why? What command did you use?

Answer:

You should have no problems. The *read* permissions are set for others on `/bin/cp`. The following command would have performed the copying function:

```
$ cp /bin/cp .
```

Advanced Exercises

1. List the names of all files in `/usr/spool` that belong to the user named `lp`. (You will have to look in find(1) for this one).

Answer:

```
$ find /usr/spool -user lp -print
/usr/spool/lp
/usr/spool/lp/class
/usr/spool/lp/interface
      :
```

2. If two files are linked, they share the same space on the disk. This space is referenced by an "inode number". Use the `ln` command to create a link between two files, and then use the `-i` option of `ls` to see that the files really have the same inode number.

Answer:

```
$ ln display1 linkfile
$ ls -li display1 linkfile
105667 -rw-rw-r--   2 bob    aeo                     22 Mar 17 1988 display1
105667 -rw-rw-r--   2 bob    aeo                     22 Mar 17 1988 linkfile
$
```

105667 is the inode number that both files share. Notice also that the link count for each file is 2.

3. You see that a file is linked to another file. How can you find the name of the other file? (Hint: There is a find(1) option that will help.)

Answer:

Find a file with the same inode number:

```
$ ls -i display1
105667  display1
```

Execute `find` using the `-inum` option to find the other names that reference this file:

```
$ find . -inum 105667 -print
./display1
./linkfile
$
```

Note that this is a simple case. It becomes a much bigger problem if files are linked in several directories.

7-5. LAB: Administration Exercises

1. Press Ctrl-g and notice the status line at the bottom of your screen.

2. Use the `write` command to save this file as a file called `another.jersey`.

Answer:

While in `vi` type `:w another.jersey`.

3. Use `:q` to exit `vi`. Why didn't you need to use `:q!`?

Answer:

You did not make any changes to the file, so `:q` let you quit.

7-6. SLIDE: Creating a New File

1. Using `vi`, edit a new file called `memo1`. Notice what creating a new file looks like.

2. Exit `vi` using `ZZ`. Did a file get created in your directory?

Answer:

NO. `vi` does not store a *new* buffer that is empty.

7-9. LAB: Moving Exercises

1. Use `vi` to edit the file called `numbers` and follow the instructions there. Also try moving down the lines by pressing (Return). Why does (Return) take you to the right on some of the blank lines?

When you are done, exit the file using `:q!`.

Answer:

(Return) takes you to the first nonblank character of the next line. The blank lines near the top of the file are truly empty, so the first nonblank character is at the beginning of the line. Near the bottom of the file, the lines are filled with spaces, so the first nonblank character is at the end of the blanks.

2. Edit the file called `new.jersey` and move to the last line.

Then, move to line 30.

Then, move to line 1.

How can you tell that you are on the correct line?

Answer:

`G` moves you to the last line, `30G` moves you to line 30, and `1G` moves you to line 1. In each case, you can use (Ctrl)-(g) to verify that you are on the correct line.

3. Move to the beginning of line 26 in `new.jersey` and type `w` five times.

Answer:

`w` stops at the punctuation marks.

4. Exit the file with `:q!`.

5. Advanced: `w` moves you forward by words. How can you move backwards by words?

Answer:

The `b` command works just like the `w` command, only backwards.

7-14. LAB: Inserting Text

1. Now, using `vi`, edit a new file called append, type an a, and then enter the following:

```
Note that typing an a while in command mode does not produce output,
but puts you into text input mode.  Now if you type commands in input mode:

aaaaaaaaaaa w w w w ZZ ZZ ZZ

they are just appended to the buffer.
```

Press Esc. Now save the contents of the buffer and leave vi.

2. Using `vi`, edit a file called `add.text`. Move to the end of the first line, type an a, a carriage (Return), and enter this line:

```
This line is being appended to my buffer using "a".
```

Then press Esc to leave input mode. Where did the text get appended in relation to the cursor?

Answer:

After the cursor. Return also gave us a new line.

3. Now, move the cursor to the first line and type an `I`. Add the following line:

```
This line is being inserted into the buffer using "I".
```

Type a carriage return and an Esc. Where does text get inserted in relation to the cursor? Save the contents of the buffer and leave `vi`.

Answer:

I inserts text at the beginning of the line, regardless of where the cursor is on that line.

4. Could you have accomplished these same tasks using `o` and `O`?

Answer:

Yes! `vi` has several ways to do everything. The more sophisticated commands usually take less work to get the same results.

5. Using `vi`, edit the file you created called append. Using only *two* commands, go to the last line and add the words `new test` to the end of the line. *Do not press* Esc *yet.*

Where was the text added?

Answer:

`G` takes you to the last line and `A` puts you in input mode at the end of that line.

6. While you are still in input mode, use the backspace key to back up over the word `test` and type `text`. What happened?

Press Esc to leave input mode. Save your changes and exit `vi`.

Answer:

The backspace key backed the cursor up over the word `test` and allowed you to overstrike it with the word `text`. Note that in input mode, the backspace key moves you to the left by one character.

7. Using `vi`, edit the file called `add.text` again. Put the cursor on the first line and type an `o` and then this line:

```
This line was opened up in my buffer using "o".
```

Then type an Esc. Where was the text added?

Answer:

On the line after the cursor.

8. Now put the cursor on the first line and type an `O`. Then add this line:

```
This line was opened in my buffer using "O".
```

Then type an Esc. Where was the text added?

Answer:

A line was opened before the line the cursor was on. The line you typed is now the first line of the file.

9. Could we have done this with an `a`?

Answer:

No, because we can only add text after the cursor with `a`.

10. Type ZZ to save the file and exit `vi`.

7-16. LAB: Deleting Text

1. Using `vi`, edit a new file called `oops`. Add the following line to the buffer;

```
This sentence contains an inncorrrrectlyy spelled word.
```

Then type an (Esc).

Now, move the cursor to the letters that do not belong and type `x`.

Answer:

The letter under the cursor is deleted.

2. Type `ZZ` to save the file and exit `vi`.

3. Using `vi`, edit the file called `delete`. Delete the two blank lines in the middle of the file with one command.

Answer:

Use `2dd`. The text will close up and redraw or will leave behind "@" characters, as you delete the lines.

4. Move the cursor to the `T` in `THIS` on the line that says `THIS LINE SHOULD BE DELETED`. Now, type `dd`. What happens?

Answer:

The line is deleted.

5. Use `u` to undo the deletion. Move the cursor somewhere else on the line and type `dd`. Do this cycle several times. Does it make a difference where the cursor is on the line?

Answer:

`dd` deletes the whole line regardless of where the cursor is on that line.

6. Type `ZZ` to save the file and exit `vi`.

7. Using `vi`, edit a file called `unwanted`. Move to line 31 and put your cursor on the string `WORD`. Type `dw`. What happens?

Answer:

The word disappears from the text.

8. Put your cursor on the string `WORD.WORD` on line 70 and type `dw`.

Answer:

Just the first WORD was deleted because w stopped at the ".".

9. Type u and then delete WORD.WORD.

Answer:

3dw would delete WORD.WORD.

10. Type ZZ to save the file and exit vi.

7-20. LAB: Modifying Text

1. Using vi, edit a file called mutant. Find the word ghouls, and change it to food.

Answer:

Using cw, the word ghouls can be changed to food.

2. Now change the word food to fools using the s command.

Answer:

Move the cursor to the d and press s. Type ls and press Esc.

3. Change the word they to this using the R command.

Answer:

Move the cursor to the e and type R. Type is and press Esc. Notice how each character you typed struck over the character under it.

4. Use a replace command to change the word rest to best.

Answer:

Move your cursor to the r in rest, press r and then b, and the change is made.

5. Type ZZ to save the file and exit vi.

7-22. LAB: Copying and Moving Text

1. Edit the file called `productivity`. Copy the title line (line 1) and place it at the end of the file so that the statement makes sense.

Answer:

On line 1, type `Y` to yank the text. Move to line 4 and type `p` to put the copy of the line down. Moving to the last line and typing `P` would also have worked.

2. The last two lines are both short. Join the last line with the one above it to make the file look better. Save the corrected file and exit `vi` when you are done.

3. Edit the file called `upside_down`. Use the technique we just learned to move the lines into their correct positions. Save the file and exit `vi`.

Answer:

For each line, you will have to delete it using `dd` and then put it where it belongs with `p` or `P`.

4. Advanced:

Using join and replace, make the lines of the file **`upside_down`** fit the margins of your screen.

Answer:

Join lines until they wrap around the right margin. Then you can replace a space near the end of the line with Return to split the line into two lines.

7-25. LAB: Searching and Replacing Exercises

1. Using `vi`, edit the file called **`new.jersey`**. Find the first instance of the word **you**.

Which way did you move in the file? (Hint: How can we tell what line we are on?)

Answer:

/you finds the first occurrence of **you** by searching downward (toward the end of the file). You can find out the line you are on by typing Ctrl-g.

2. Type `n` once. Where did the cursor go, and in which direction did it travel to get there?

Answer:

It went to the next instance of **you** by searching toward the end of the file.

3. Go back to line 1 and search for the first occurrence of the word `the`. What happened?

Answer:

You find `them` instead because `vi` is just looking for the three characters `the` wherever they appear.

4. Type `:q` to exit `vi`. Why did it let you out?

Answer:

If you do not make changes, `vi` will let you exit a file without saving it first.

5. Using `vi`, edit the file `new.jersey` again. Change all occurrences of `like` to `hate`.

Answer:

Type `:1,$s/like/hate/g`

8-7. LAB: Exercises

1. Make sure you are in your home directory. What happens when you type `more n` Esc Esc? Using this command line, how can you make it display `new.jersey`.

Answer:

Typing the command line given puts `more n` on the command line, and the K-shell beeps because there is more than one file starting with `n`. If you type an `e` and then Esc Esc again, the file name `new.jersey` will be completed for you.

2. Create an alias called "psf" that executes the `ps -ef` command.

Answer:

```
$ alias psf='ps -ef'
```

3. Edit your `.profile` to set the *ENV* variable to `~/.kshrc`.

Advanced:

Do this from the directory you are in using the ~ in the path name.

Answer:

You would add these two lines to your `.profile`:

```
ENV=~/.kshrc
export ENV
```

4. Check the commands in the `.kshrc` file in your home directory. Add your psf alias to the list.

5. Log out and then log back in to test your aliases. Why did you have to log out?

Answer:

You had to reread the `.profile` and `.kshrc` files. The easiest way is to log out and then log back in.

6. Type this incorrect command without hitting (Return):

```
cd /user/spol/ko/interface
```

Using command line editing, correct the line to read:

```
cd /usr/spool/lp/interface
```

(Do *not* retype the command)

Answer:

```
$ cd /user/spol/ko/interface(Esc)
```

Using (Backspace) and the space bar to position the cursor, use `vi` commands `x`, a, `cw` to make the appropriate changes. Remember to use (Esc) whenever you need to leave input mode.

7. Execute the command `ls -F ~/cars+dogs`.

Recall this command line and change the `ls -F` to `ls -l` using whatever `vi` editing commands are necessary. Re-execute the command.

Advanced:

Try the command `ls -F ~/cars+dogs` without typing `cars+dogs` but using file name completion.

Answer:

```
$ ls -F ~/cars+dogs
car.models/    collie    dog.breeds/    poodle
$ (Esc) (k)
```

Now use the `r` command to change `ls -F` to `ls -l` and press (Return).

```
$ ls -F ~/ca (Esc) (Esc)
```

9-3. SLIDE: Input Redirection — <

1. What is the difference between `more new.jersey` and `more < new.jersey`? Try both and notice the slight difference.

Answer:

In the command line `more new.jersey`, `more` is reading the file `new.jersey`, while in `more < new.jersey`, `more` is reading stdin, but we have redirected input from `new.jersey`

In the first case, `more` knows the file name and size, so it can print a percentage at the bottom of each screen. When `more` reads stdin, it has no idea how large its input is, so it just displays one screen at a time until the input stops. Because it does not know how large the input is, it cannot display the percentage at the bottom of each page.

9-10. LAB: Exercises

1. Create a long listing of your home directory and save it in a file called `myfiles`.

Answer:

```
$ cd
$ ls -l > myfiles
$
```

2. Use output redirection to create a file that contains:

- Today's date and time.
- Your current directory.
- A list of who is on the system.

Answer:

```
$ date > file
$ pwd >> file
$ who >> file
```

3. Use a pipeline to display a long listing of the `/etc` directory one screen at a time.

Answer:

```
$ ls -l /etc | more
```

4. Construct a pipeline to determine how many files are in your home directory.

Answer:

```
$ ls | wc -l
     18
```

Advanced:

5. `sort` is a command that reads stdin, sorts the lines of input, and then writes the sorted lines to stdout. If you give it a file name as an argument, it will sort the lines of the file and write the result on stdout without ever changing the input file.

Use a pipeline to produce a sorted listing of the users on the system.

Answer:

```
$ who | sort
chip       tty1p4       Jul 25 06:43
frank      tty1p5       Jul 27 07:42
ralph      tty2p0       Jul 27 09:09
linda      tty3p2       Jul 25 16:07
```

10-7. LAB: Exercises

1. List the current status of the printers in the lp spooler system and find the name of the default printer.

Answer:

```
$ lpstat -t
scheduler is running
system default destination: rw
device for rw: /dev/rw
rw accepting requests since Tue Jun 04 10:56:20 1991
printer rw is idle.  enabled since Thu Jun 27 14:32:52 1991
        fence priority : 0
```

2. Send the file named `new.jersey` to the line printer. Make a note of the request id that is displayed on your terminal.

Answer:

```
$ lp new.jersey
request id is rw-58 (1 file)
```

3. Using a pipe, send the output of a long listing of your home directory to the printer. Submit the request so it will print with the title "My files" on the banner page.

Answer:

```
$ ls -l | lp -t"My files"
request id is rw-59 (standard input)
```

4. Verify that your requests are queued to be printed.

Answer:

```
$ lpstat
rw-58              ralph            3967   Thu Jul 04 12:57:25 1991
rw-59              ralph            1331   Tue Jul 02 13:01:19 1991
```

5. How can you tell what files other users are printing? Try it.

Answer:

You can tell by using `lpstat -t`.

6. Use the `cancel` command to remove your requests from the line printer system queue. Confirm that they were canceled.

Answer:

```
$ cancel rw-58  rw-59
request "rw-58" canceled
request "rw-59" canceled
$ lpstat
$
```

7. Advanced: pr is a command that formats output for a printer. It will print a header on each page which includes the date, file name, and page number. Try to pr the file `new.jersey`. (You may need to pipe it to `more` to see the output). How could you use pr with `lp`?

Answer:

```
$ pr new.jersey | more
Dec 11 13:03 1990  new.jersey Page 1


                    I LIKE NEW JERSEY BECAUSE...

$ pr new.jersey | lp
request id is lp-355 (standard input)
```

11-6. LAB: Exercises

1. Modify your `.profile` to include these changes:

- Add `/usr/widgets/bin` to the *PATH*.
- Add the `bin` directory under your home directory to the *PATH*.
- Create a variable named *WIDDIR* and set its value to `/usr/widgets`.

Advanced:

How can you add the `bin` directory to the *PATH* without typing the full path name to your home directory on the line?

Answer:

The modified `.profile` should have lines that look something like this:

```
# Set up the search paths:
PATH=.:/bin:/usr/bin:/usr/contrib/bin:/usr/local/bin
PATH=$PATH:/usr/widgets/bin:$HOME/bin

# Set up the shell variable:

WIDDIR=/usr/widgets
export WIDDIR
```

2. Have the new variables from the last exercise changed in your environment yet? Do whatever is necessary to make these changes active. Use the **env** command to make sure the changes were made correctly.

Answer:

No, the variables in `.profile` are only set when you log in to the system. Therefore, you must log out and log back in. The **env** output should look something like this:

```
_=/bin/env
WIDDIR=/usr/widgets
HOME=/users/gerry
PWD=/users/gerry
SHELL=/bin/ksh
MAIL=/usr/mail/gerry
EDITOR=vi
LOGNAME=gerry
TERM=70092
ENV=/users/gerry/.kshrc
PATH=.:/bin:/usr/bin:/usr/contrib/bin:/usr/local/bin:/usr/widgets/bin
TZ=EST5EDT
```

3. Temporarily set your terminal type to "unknown" (Do not put this change in the `.profile`.) Use **env** to make sure it is set correctly.

Try to edit the file `new.jersey` using `vi`. What happened?

Answer:

To set *TERM*, use the command `TERM=unknown`.

When you try to run `vi`, it enters a line mode called **open mode** because the terminal type is not set correctly. The `vi` commands still work in open mode, but you only see one line at a time instead of a full screen.

4. Exit `vi` and set your *TERM* variable to normal without logging out.

Use the `tset` command to reset the terminal characteristics.

Answer:

To set the terminal type back, type `TERM=`*type*, where *type* is the terminal type listed in the terminal type table shown earlier.

Advanced Exercise:

5. *PS1* is a shell variable that is used as your shell prompt. Change the value of *PS1* so your prompt is `What now?` .

Answer:

```
$ PS1="What now? "
What now?  whoami
gerry
What now?
```

A-7. LAB: Exercises

1. Execute the ps command to show all processes running on the system. How many processes are running? (Use a pipe!) How many processes are you running?

Answer:

```
$ ps -ef
     UID   PID  PPID  C   STIME TTY      TIME COMMAND
    root    38     1  0  Jul 25 ?        0:13 /etc/cron
    root    97     1  0  Jul 25 console  0:00 /etc/getty console console
    root     4     0  0  Jul 25 ?        2:11 netisr
    root     3     0  0  Jul 25 ?        0:02 statdaemon
    root     2     0  0  Jul 25 ?        0:00 pagedaemon
    root     1     0  0  Jul 25 ?        0:07 init
    root     0     0  0  Jul 25 ?        0:04 swapper
    root   103     1  0  Jul 25 tty1p0   0:00 /etc/getty -h tty1p0 9600
    root  1215     1  0  Jul 26 tty1p1   0:00 /etc/getty -h tty1p1 9600
    bill   116     1  0  Jul 25  tty3p2   0:06 -ksh [ksh]
    bill  1932   116  1 10:54:36 tty3p2   0:07 vi mod11
    bill  1994  1932 70 11:39:24 tty3p2   0:03 ps -ef
.
.
       etc
```

You can see every process that is running. Simply count them to find out how many there are. The pipe to count processes could be `ps -ef | wc -l`. This would also count the header line.

Look through the ps listing to find processes with your name in the UID column.

2. Looking at the listing, can you tell what PPID means?

Answer:

PPID is parent process id. This is the process that started the process you are looking at.

3. If you are running `vi` and your terminal locks up, how can you fix it?

Answer:

Log in to another terminal, execute ps to find the PID of the hung `vi` process, and execute the `kill` PID command to kill it.

4. Who are the only people who can properly shut down the system?

Answer:

The root (the superuser) and/or anyone who is given authorization by being included in the `shutdown.allow` file.

5. What does the **-h** option to the **shutdown** command do?

Answer:

It halts the system.

6. Why is it important that the system be quiet when you shut it down?

Answer:

The more activity there is on a system, the greater chance you have of losing your work when the system goes down.

7. Write (*don't execute*) the command to shut down the system to a halt in 30 seconds.

Answer:

```
# shutdown -h 30
#
```

Notice that the prompt is a "#". This is the prompt the superuser gets when he or she logs in.

B-4. SLIDE: Using Visible Softkey Commands

For each of the following exercises, use the *visible* softkeys and the option softkeys to enter the command.

1. Get a long listing of the contents of your login directory. Look at the translation of the softkeys before executing the command.

Answer:

Press `List files` from the first bank of softkeys. Press `long format` from the option softkey menu. Press (Insert Line) to see the translation and then press (Return).

2. Change the permission of the file `mutant` so you have read and write permission and everyone else just has read permission. Execute the command immediately.

Answer:

Toggle the `--More--` key to get to the second bank of softkeys, and press `Set file attribs`. Press `mode` from the option softkey menu, then press `readable by`, `all`, `writable by`, and `owner` in this order. Notice that each time you press a key from the option softkey menu, a new option softkey menu appears. Finally, type the name of the file, `mutant`, and press (Return).

B-5. SLIDE: Using Invisible Softkey Commands

For each of the following exercises, use the *invisible* softkey commands and the option softkeys.

1. Using the `who` *invisible* softkey, list those login lines that are not currently being used. Execute the command immediately.

Answer:

Type `who` at the `$` prompt. When the softkey options appear, press `idle tty lines`; then press (Return).

2. Find out how many lines are in the file `new.jersey`. Look at the the translation of the softkeys before executing the command.

Answer:

Type `wc` at the `$` prompt. When the softkey options appear, press `lines`; then type the name of the file, `new.jersey`. Finally, press (Insert Line) to see the translation and then press (Return)

B-8. LAB: Exercises

Complete each of the following exercises using the Key Shell soft keys.

1. Notice the two files named `display1` and `display2` in your home directory. Print their contents to your terminal.

Answer:

Press `Display files` and then type `display1` and `display2`.

2. Make a copy of `display1` and call it "More_Than_14_chars". What happened?

Answer:

Press `Copy files` from the second bank of softkeys. Then type `display1`, press `to`, and type `More_Than_14_chars`. Return to the first bank of softkeys and press `List files`. You will notice that the file was truncated to 14 characters unless you are using long file names. If you are on system using long file names, the complete file name will be used.

3. Rename `display1` to `new.display1`.

Answer:

Press `Move files` from the second bank of softkeys. Then type `display1`, press `to`, and type `new.display1`.

4. Construct an HP-UX command that will copy `display2` to `new.display1` but will prompt you to verify that the file should be copied when copying it would overwrite an existing file.

Answer:

Press `Copy files` from the second bank of softkeys, then press `interactively` from the option softkeys that appear. Type `display2`, press `to`, and type `new.display1`. You should see the prompt `overwrite display2? (y/n)`.

5. Construct an HP-UX command that will find all files in your home directory that are larger than 1200 bytes.

Answer:

Press `Find files` from the second bank of softkeys, type the name of your home directory, press `size` from the second bank of option softkeys, press `greater than`, and type 1200.

6. Construct an HP-UX command that will display all the processes running on the system that belong to another member of the class. (You will need to know how that person logged in.)

Answer:

Press `Process info` from the fourth bank of softkeys, press `for user` from the option softkeys that appear, and then type the name of the person for whom you are displaying running processes.

C-3. SLIDE: Using the Workspace Manager

1. Start two terminal windows in your current workspace.

Answer:

Click twice on the terminal window on the front panel.

2. Switch to another workspace and start a terminal window there.

Answer:

Click on another workspace, then click on the terminal window.

3. Toggle between your two workspaces.

Answer:

Click on the two workspaces you have activated.

4. Using the Mail icon on the front panel, mail your partner one of your files.

Answer:

Press and hold down the middle mouse button on a file in the file manager, drag the file over the mail icon, release the middle mouse button. Position the pointer in the displayed dialog box so that you can type the destination and an optional subject, press (Return).

5. Print a file in your home directory by using the Printer icon.

Answer:

Drag a file from the file manager down to the print icon.

C-4. SLIDE: Working with Terminal Windows

1. Create two terminal windows in your workspace.

Answer:

Click twice on the terminal button on the front panel.

2. Move one of the terminal windows to a different location in your workspace.

Answer:

Move the pointer to the title bar, press and hold the left button, drag the window to a new location.

3. Iconify the other terminal window and then restore it to its normal size.

Answer:

Click on the iconify button. To restore the icon to its normal size, double click on the icon, or click the left button on the icon to bring up the window menu, and choose `Restore`.

4. Make one of the windows bigger using the mouse and frame edge.

Answer:

Move the pointer to a place on the window frame, press and hold down the left button, drag the frame to its new size, and release the button.

5. Activate the other terminal window and execute the `ls -l` command to see that it works just like a text terminal.

Answer:

Click on the other window and execute `ls -l`.

6. Close one of the windows.

Answer:

Double click on the window menu button.

C-5. SLIDE: The File Manager

1. Create a new file manager window for the directory cars+dogs that is in your current file manager window.

Answer:

Move the pointer over the directory icon for cars+dogs. Press and hold down the middle mouse button, drag the icon over the workspace background, and release the middle mouse button.

2. Display the contents of your home directory from the new file manager you created in exercise 1.

Answer:

Double-click your home directory path segment on the status line in the file manager for cars+dogs.

C-6. SLIDE: Manipulating Files

1. Create files new.one and new.two in the directory cars+dogs.

Answer:

Double click on cars+dogs to open a file manager view of this directory. Choose New ... from the File menu, click the New Data File Name box, type new.one, press Return. Repeat the process for new.two.

2. Rename new.one to new.old.

Answer:

Click new.one, choose Rename ... from the File menu, position pointer at end of new.one in the New File Name box, click, back space as necessary, type new.old, press Return.

3. Copy new.two to your parent directory.

Answer:

Click the file manager button in the workspace manager to open another File Manager view of your home (parent) directory. Move your pointer over new.two, press and hold down Ctrl, press and hold down the middle mouse button. Drag new.two over an empty area in the parent directory, release the middle mouse button, and release Ctrl.

4. Move new.old to your parent directory.

Answer:

Press and hold down the middle mouse button on new.old, drag new.old over an empty area in the parent directory, and release the middle mouse button.

5. Remove new.two and new.old from your parent directory.

Answer:

Press and hold down the middle mouse button on `new.two` in the parent directory, drag the file over the trash can, and release the middle mouse button. Repeat the process for `new.old`.

6. Restore `new.old` to your parent directory.

Answer:

Click the Trash Can in the workspace manager, click `new.old` in the Trash Can list area, choose `Restore` from the `Edit` menu, and choose `Exit` from the `File` menu.

C-7. SLIDE: Manipulating Directories

1. Create directories `DIRECT` and `SHOW` in the directory `cars+dogs`.

Answer:

Choose `New ...` from the `Directory` menu, click the `New directory Name` box, type `DIRECT`, press Return. Repeat the process for `SHOW`.

2. Rename `DIRECT` to `REDIRECT`.

Answer:

Click `DIRECT`, choose `Rename ...` from the `File` menu, position pointer at beginning of `DIRECT` in the `New File Name` box, click, type `REDIRECT`, and press Return.

3. Move `SHOW` to your parent directory.

Answer:

Press and hold down the middle mouse button on `SHOW`, drag `SHOW` over an empty area in the parent directory, and release the middle mouse button.

4. Remove `SHOW` from your parent directory.

Answer:

Press and hold down the middle mouse button on `SHOW` in the parent directory, drag the file over the trash can, and release the middle mouse button.

5. Restore `SHOW` to your parent directory.

Answer:

Click the Trash Can in the workspace manager, click `SHOW` in the Trash Can list area, choose `Restore` from the `Edit` menu, and choose `Exit` from the `File` menu.

C-8. SLIDE: Using the Help Manager

1. Open the Help Index window from the front panel of the workspace manager; display information on the topic Help on the Workspace Manager.

Answer:

Click the help button, [?], on the front panel of the workspace manager, choose Help on the Workspace Manager in the Help Index, and click Viewer from the Help Index.

2. Go to a File Manager window and display the Help Viewer window for this application.

Answer:

Click Help in the file manager menu bar,and choose On Application ... .

3. Go to a File Manager window and bring up a dialog box; get help on this dialog box.

Answer:

Click File in the file manager menu bar, choose new ... , and click Help in the dialog box.

C-9. SLIDE: Using the Style Manager

1. Change your workspace colors.

Answer:

Click the Color button in the style manager, choose a color, and click [OK].

2. Change the backdrop pattern of one of your workspaces.

Answer:

Click the Backdrop button in the style manager, choose a backdrop, click [Apply], click [Close].

3. Change the font size to large.

Answer:

Click the fonts button in the style manager, choose Large from the Font Size option menu, click [OK] in the Notice dialog box, click [OK] in the fonts dialog box, select Resume current session in the Session Startup dialog box, logout, and log back in.

D-9. LAB: Exercises

Directions:

Ask your instructor which exercises you can do in the classroom. Also find out the host names of the computers you can communicate with.

1. Use the hostname command to determine the name of your local system. What systems can you communicate with?

Answer:

The `hostname` command reports the local host name. By looking at the `/etc/hosts` file, you can see all of the computers your local computer can talk to.

2. Use `telnet` to log in to another computer. Use the `hostname` command to verify that you are connected to the correct computer. Log off the remote computer when you are done.

Answer:

```
$ telnet fred
Trying...
Connected to fred
Escape character is '^]'.

HP-UX fred 8.00 B 9000/350

login:  enter your name
Password: enter your password
      .
      .
      .
$ hostname
fred
$ exit
```

3. Transfer one of your files to your home directory on a remote computer using `ftp`, and then use `rcp` to copy another file to the remote machine. Notice the differences.

Answer:

In `ftp`, you would use the `put` command, similar to the example given in the student notes.

4. Use `remsh` to list the contents of the remote directory to verify that the copy worked.

Answer:

```
$ remsh system ls
```

The `ls` command will list your *home* directory on *system*.

** For HP Internal Reference Only **

Manufacturing Part Number

51489-90011

Customer Order Number

NONE

Printed in USA