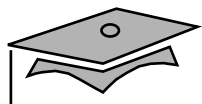


Fundamentals of Solaris™ 8 Operating Environment for System Administrators

SA-118





Copyright 2000 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, California 94303, U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun Logo, Solaris Operating Environment, Common Desktop Environment (CDE), admintool, SPARC, and SunOS are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

U.S. Government approval required when exporting the product.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g) (2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015 (b)(6/95) and DFAR 227.7202-3(a).

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS, AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.



About This Course



Course Goal

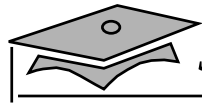
This *Fundamentals of Solaris™ 8 Operating Environment for System Administrators* course teaches you how to use UNIX® operating system commands and basic Solaris™ Operating Environment commands.



Course Overview

The class is for new users of the Solaris Operating Environment. You will learn fundamental command-line features of the Solaris Operating Environment, including:

- Directory tree navigation
- File permissions
- The `vi` text editor
- Command shells
- Basic network use



Course Map

Getting Started

"Introduction to
the Solaris 8
Operating
Environment"

"Accessing the
System"

File Operations

"Accessing Files
and Directories"

"Directory and
File
Commands"

"Searching for
Files and Text"

"File Security"

Text Editing

"Visual (vi)
Editor"

Saving Data

"Archiving User
Data"

Connecting to Other Hosts

"Remote
Connections"

Manipulating Processes

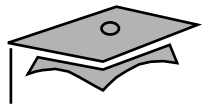
"System
Processes"

Shell Operations

"The Korn
Shell"

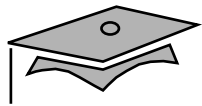
"Introduction to
sed and awk"

"Reading Shell
Scripts"



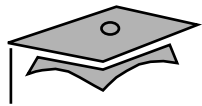
Module-by-Module Overview

- Module 1 – “Introduction to the Solaris 8 Operating Environment”
- Module 2 – “Accessing the System”
- Module 3 – “Accessing Files and Directories”
- Module 4 – “Directory and File Commands”
- Module 5 – “Searching for Files and Text”
- Module 6 – “File Security”
- Module 7 – “Visual (vi) Editor”



Module-by-Module Overview

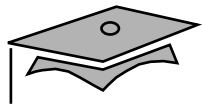
- Module 8 – “Archiving User Data”
- Module 9 – “Remote Connections”
- Module 10 – “System Processes”
- Module 11 – “The Korn Shell”
- Module 12 – “Introduction to sed and awk”
- Module 13 – “Reading Shell Scripts”



Course Objectives

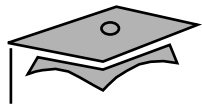
Upon completion of this course, you should be able to:

- Log in and log out of the Solaris Operating Environment and CDE systems
- Compose command-line strings to perform functions
- Navigate the Solaris Operating Environment directory tree
- Manipulate text files
- Create files and directories
- Back up and restore user files and directories



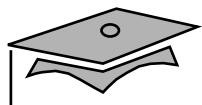
Course Objectives

- Change permissions of files and directories
- Use the `vi` text editor
- Identify and modify initialization files
- Employ shell features to streamline command execution
- Use basic network commands
- Use commands to search directories and files
- List active user processes and selectively kill user processes



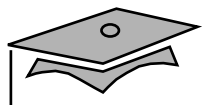
Skills Gained by Module

Skills Gained	Module												
	1	2	3	4	5	6	7	8	9	10	11	12	13
Log in and out of the Solaris Operating Environment and CDE systems													
Compose command-line strings to perform the Solaris Operating Environment functions													
Navigate the Solaris Operating Environment directory tree													
Create files and directories													
Manipulate text files													
Use commands to search directories and files													
Change permissions of files and directories													
Use the vi text editor													
Back up and restore user files and directories													
Use basic network commands													
List active user processes and selectively kill user processes													
Employ shell features to streamline command execution													
Identify initialization files and read basic scripts													



Guidelines for Module Pacing

Module	Day 1	Day 2	Day 3	Day 4
"About This Course"	A.M.			
"Introduction to the Solaris 8 Operating Environment"	A.M.			
"Accessing the System"	A.M. /P.M.			
"Accessing Files and Directories"	P.M.			
"Directory and File Commands"		A.M.		
"Searching for Files and Text"		A.M.		
"File Security"		P.M.		
"Visual (vi) Editor"		P.M.		
"Archiving User Data"			A.M.	
"Remote Connections"			A.M. /P.M.	
"System Processes"			P.M.	
"The Korn Shell"				A.M.
"Introduction to sed and awk"				A.M. /P.M.
"Reading Shell Scripts"				P.M.



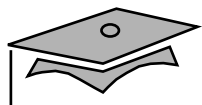
Topics Not Covered

- Detailed shell programming:

Covered in SA-225: Advanced UNIX® Utilities and Shell Programming for Administrators

- System administration concepts:

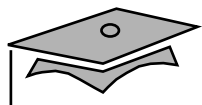
Covered in SA-238: Solaris™ 8 Operating Environment System Administration I



How Prepared Are You?

To be sure you are prepared to take this course, can you answer yes to the question below?

- Can you input commands and control characters from a keyboard?



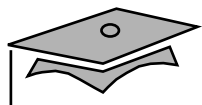
Introductions

- Name
- Company affiliation
- Title, function, and job responsibility
- Solaris Operating Environment experience
- Reasons for enrolling in this course
- Expectations for this course



How to Use the Course Materials

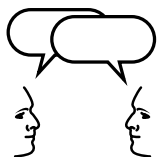
- Course Map
- Objectives
- Lecture
- Exercises
- Check Your Progress



How to Use the Icons



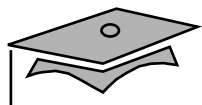
Additional resources



Discussion



Exercise objective



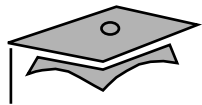
How to Use the Icons



Caution

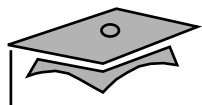


Warning



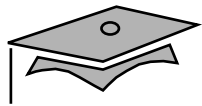
Typographical Conventions and Symbols

- `Courier` is used for the names of command, files, and directories, as well as on-screen computer output.
- **`Courier bold`** is used for characters and numbers that you type.
- *`Courier italic`* is used for variables and command-line placeholders that are replaced with a real name or value.
- *Palatino italics* is used for book titles, new words or terms, or words that are emphasized.



Module 1

Introduction to the Solaris 8 Operating Environment



Objectives

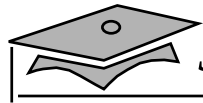
Upon completion of this module, you should be able to:

- List the four main hardware components of a computer
- Describe the four main components of the Solaris Operating Environment
- State the three main components of the SunOS™ operating system
- Identify the default shells available in the Solaris Operating Environment
- Understand the basics of the Common Desktop Environment (CDE)

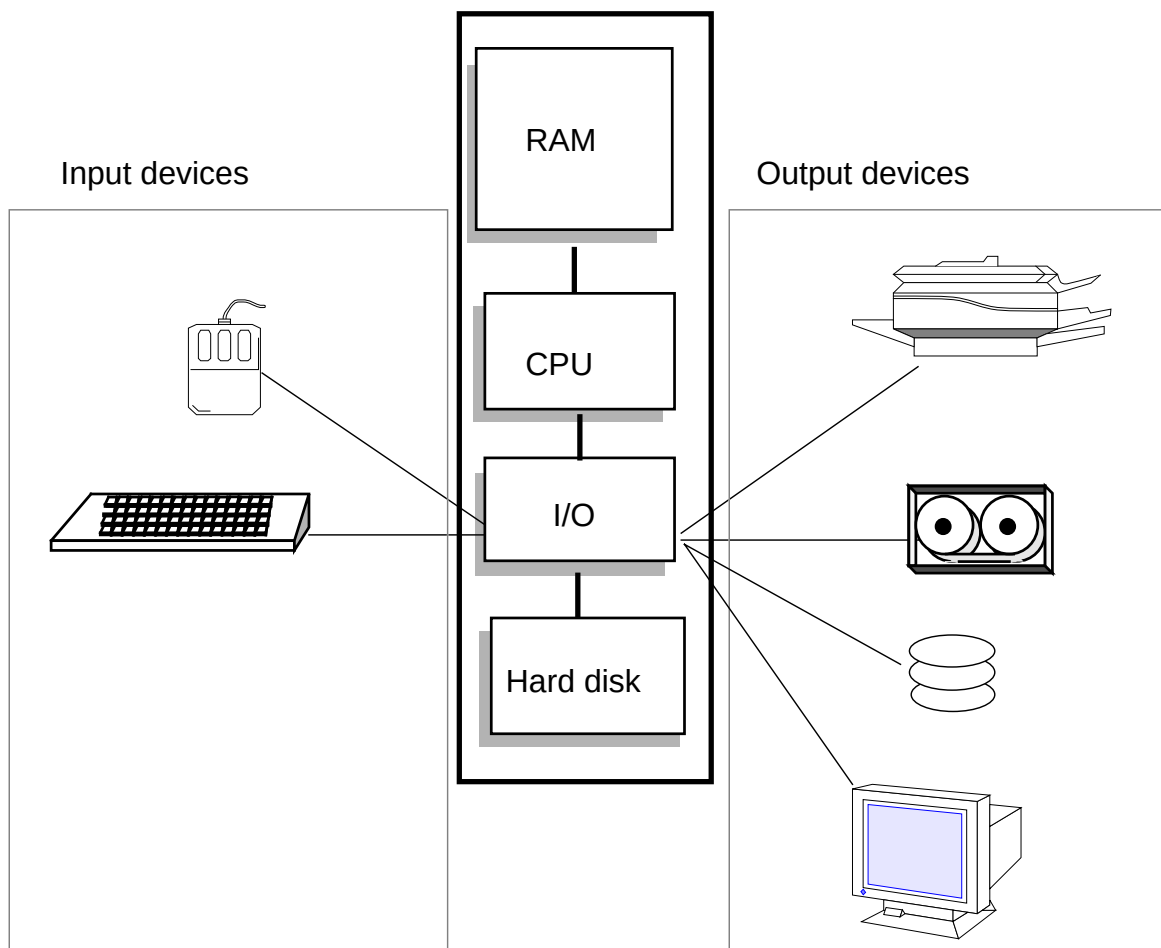


Main Hardware Components of a Computer

- Random access memory (RAM)
- Central processing unit (CPU)
- Input/output (I/O)
- Hard disk or other mass storage device



Main Hardware Components of a Computer

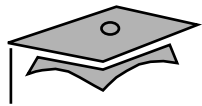




The Solaris Operating Environment

The Solaris Operating Environment contains the following four components:

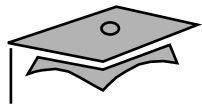
- SunOS 5.x operating system
- ONC+™ Technologies (Open Network Computing)
- Solaris Common Desktop Environment (CDE) graphical user interface (GUI)
- OpenWindows™ graphical environment



The SunOS Operating System

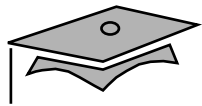
There are three main components to the SunOS operating system:

- The kernel
- The shell
- The directory tree

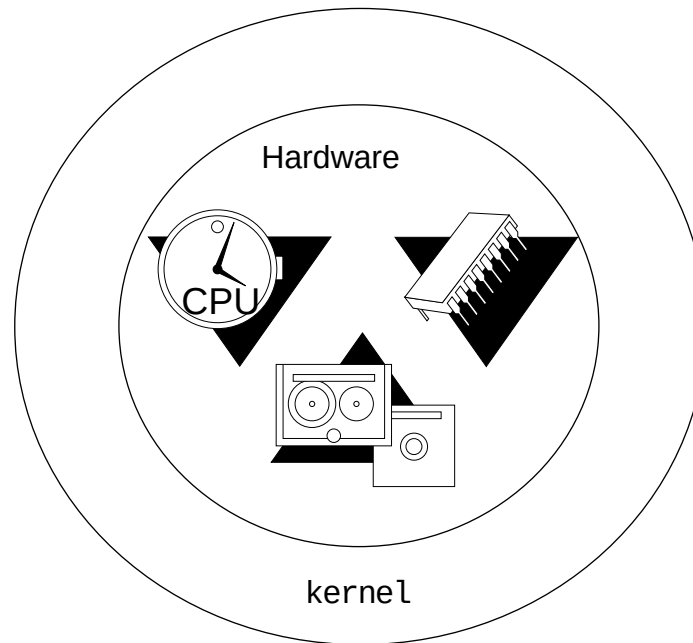


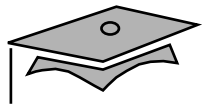
The kernel

- The core of the SunOS operating system
- Manages all the physical resources of the computer, including:
 - ▼ File systems
 - ▼ Device management
 - ▼ Process management
 - ▼ Memory management



The kernel



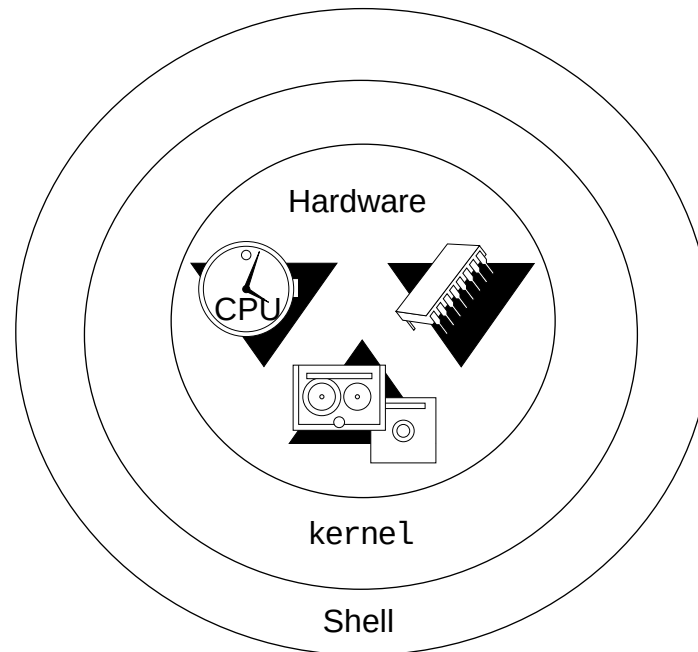


The Shell

- An interface between the user and the kernel
- The Solaris Operating Environment supports three primary shells:
 - ▼ Bourne shell
 - ▼ C shell
 - ▼ Korn shell



The Shell





The Bourne Shell

- The original UNIX shell
- The default shell used in the Solaris Operating Environment
- Default prompt is a dollar sign (\$)



The C Shell

- Developed by Bill Joy at University of California at Berkeley
- Based on the C programming language
- Default prompt is a host name and percent sign (*hostname%*)



The Korn Shell

- A superset of the Bourne shell
- Recommended as standard shell for ordinary system users
- Default prompt is a dollar sign (\$)



Exercise: Using the Solaris 8 Operating Environment

- Objectives
- Tasks
- Discussion
- Solutions



Check Your Progress

Check that you are able to accomplish or answer the following:

- List the four main hardware components of a computer
- Describe the four main components of the Solaris Operating Environment
- State the three main components of the SunOS operating system
- Identify the default shells available in the Solaris Operating Environment



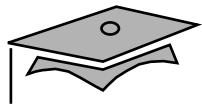
Check Your Progress

- Have a basic understanding of the Common Desktop Environment (CDE)



Module 2

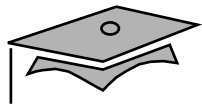
Accessing the System



Objectives

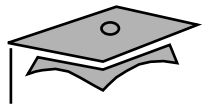
Upon completion of this module, you should be able to:

- List the characteristics of an effective password
- Log in and log out of a system from the command line
- Log in and log out of a CDE session
- Execute basic commands
- Change your password
- Identify and describe the components of a command line



Objectives

- Use control characters to erase a command line, stop the execution of a command, and stop and start screen output
- Display the online manual pages
- Search the online manual pages by keyword
- Identify users logged in to the system using the commands `who`, `who am i`, and `id`
- Enter multiple commands on a single command line



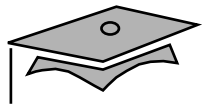
User Accounts

- The root account
- The `/etc/passwd` entry

Placeholder	UID	Comment
user1	x:102:10	User Account 1
		/export/home/user1:/bin/ksh

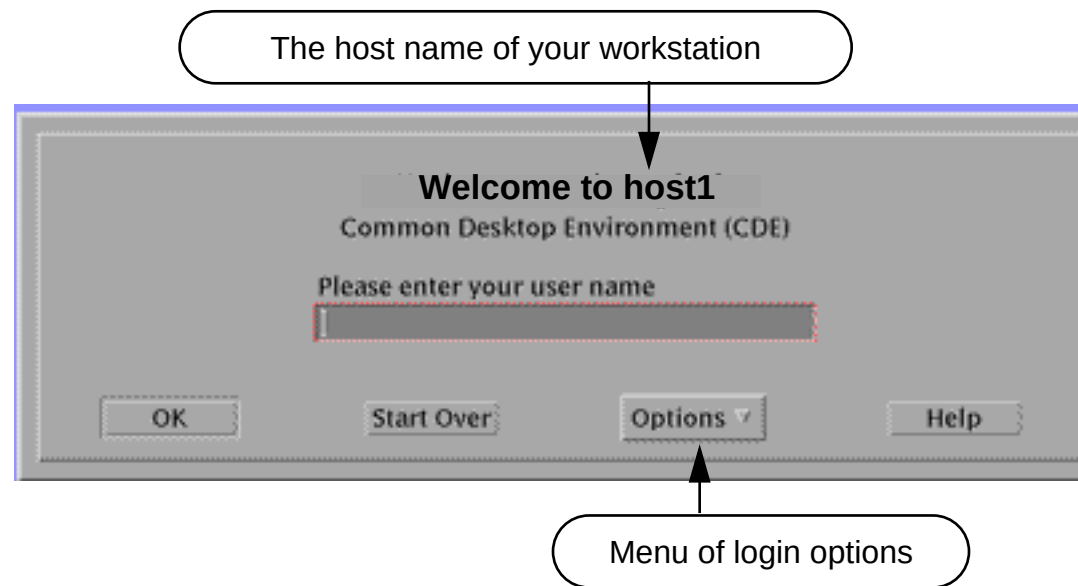
Diagram illustrating the fields of a user account entry in the `/etc/passwd` file:

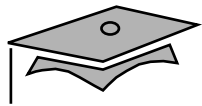
- `user1`: Login ID
- `x`: Placeholder
- `102`: UID
- `10`: GID
- `User Account 1`: Comment
- `/export/home/user1`: Login directory
- `/bin/ksh`: Default shell



Logging In

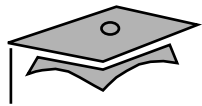
- The log in process identifies a user to the system





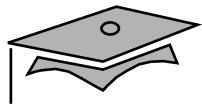
Logging In Using the Login Screen

1. Enter your user name in the text field, then press the Return key or click on the OK button.
2. You must then enter your password in the password text field and press the Return key or click on the OK button.



Logging In Using the Command Line

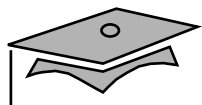
1. Display the Options pull-down menu and select Command Line Login. The Login screen disappears and is replaced by a console prompt.
2. Press the Return key to get a prompt for a user name entry.
3. Enter your user name (or login ID) at the prompt and press the Return key.
4. Enter your password at the prompt and press the Return key.



Password Requirements

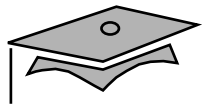
Passwords protect user accounts from unauthorized access. A user's password:

- Must be six to eight characters in length
- Should contain at least two alphabetic characters and must contain at least one numeric or special character
- Must differ from the login ID name
- Must differ from the previous password by at least three characters
- Can contain spaces



Changing Your Password

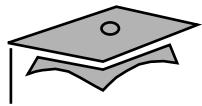
- Frequently changing user passwords helps prevent unauthorized access to the system.
- You can change your password in a CDE session or from the command line.



Securing Your CDE Session

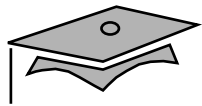
Securing your CDE session prevents unauthorized users from gaining access to the system. The two ways to secure the system are:

- Locking the screen
- Exiting the session:
 - ▼ Use the Exit button
 - ▼ Use the log out option from the Workspace Menu



Basic UNIX Commands

- The `uname` command lists information about the system.
- The `date` command displays the system's current date and time.
- The `cal` command displays a calendar for the current month and year.



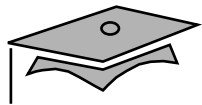
Basic UNIX Commands

- The basic syntax of a UNIX command includes:

\$ command option(s) argument(s)

- For example:

<i>\$ cal 2000</i>	(Command and argument)
<i>\$ uname -rpns</i>	(Command and multiple options)
<i>\$ cal 12 1999</i>	(Command and two arguments)
<i>\$ banner "hi there"</i>	(Command and argument)
<i>\$ ls</i>	(Command)
<i>\$ ls -l</i>	(Command and option)
<i>\$ ls directory_name</i>	(Command and argument)
<i>\$ ls -l directory_name</i>	(Command, option, and argument)



Control Characters

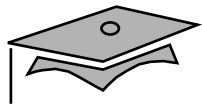
- By using some special control characters, you have added capabilities from the keyboard.
- Hold down the Control key and press the appropriate character on the keyboard for the desired action.

Control Characters	Purpose
Control-c	Terminates a command
Control-d	Indicates end-of-file or exit
Control-q	Resumes screen output if the screen is locked
Control-s	Stops screen output
Control-u	Erases the command line
Control-w	Erases the last word on the command line



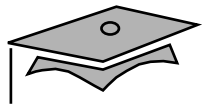
Viewing Online Documentation

- The online UNIX Reference Manuals (also called *man pages*) provide detailed descriptions about commands and their usage.
- Command format: `$ man command_name`



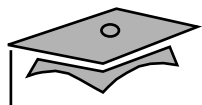
Determining Current Users

- The `who` command displays a list of users currently logged on to the local system including:
 - ▼ Login name
 - ▼ Login TTY port
 - ▼ Login date and time
 - ▼ Elapsed time since their last activity
- Command format: `who`



Identifying the User Name

- Use the `who am i` command to identify the login user name.
- Command format: `who am i`



Identifying User Group Details

- Use the `id` command to identify the user ID, user name, group ID, and group name of a system user.
- Command format: `id username`



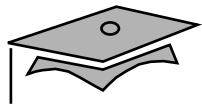
Entering Multiple Commands from a Single Command Line

- The semicolon (;) is a special character often referred to as the *command separator*.
- The semicolon gives you the ability to enter multiple commands on a single command line. The shell executes each command from left to right when the Return key is pressed.



Exercise: Accessing the System

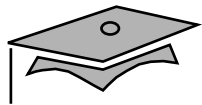
- Objectives
- Tasks
- Discussion
- Solutions



Check Your Progress

Check that you are able to accomplish or answer the following:

- List the characteristics of an effective password
- Log in and log out of a system from the command line
- Log in and log out of a CDE session
- Execute some basic commands
- Change your password
- Identify and describe the components of a command line



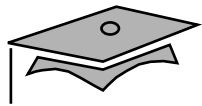
Check Your Progress

- Use control characters to erase a command line, stop the execution of a command, and stop and start screen output
- Display online manual pages
- Search the online manual pages by keyword
- Identify users logged on to the system using the `who`, `who am i`, and `id` commands
- Enter multiple commands on a single command line



Module 3

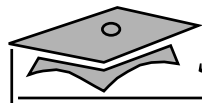
Accessing Files and Directories



Objectives

Upon completion of this module, you should be able to:

- Demonstrate the difference between absolute and relative pathnames
- Access files and directories within the directory tree using absolute and relative pathnames
- Use pathname abbreviations to access files and directories within the directory tree
- List the contents of directories and determine file types
- Identify various shell metacharacters to abbreviate file names and pathnames

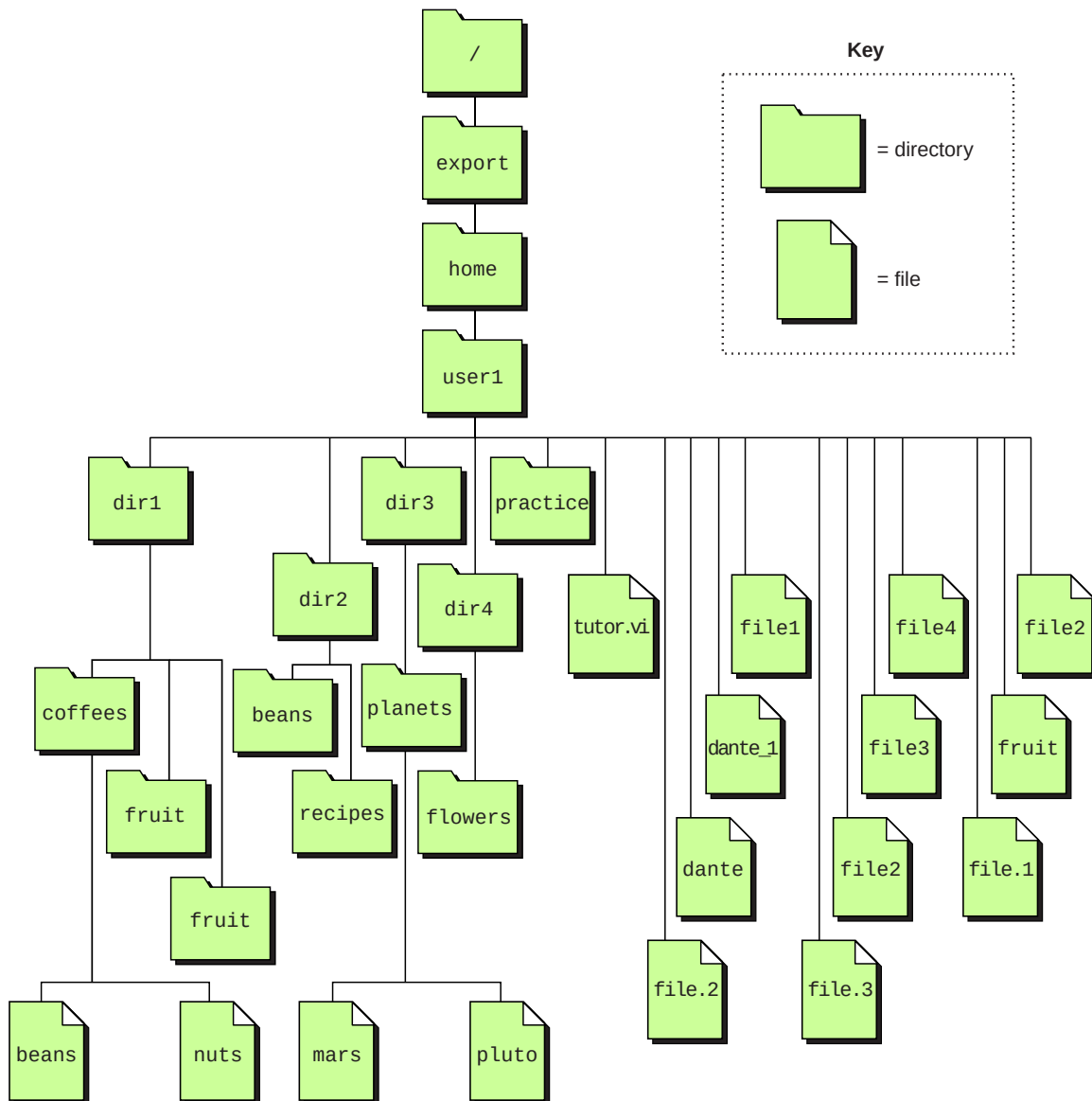


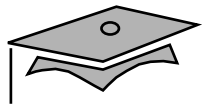
The Directory Tree

SA118_001.eps

5/24/00 jr

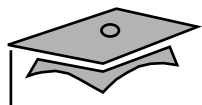
ARTIST NOTE:
Adjust graphic to top and flush left of bounding box.



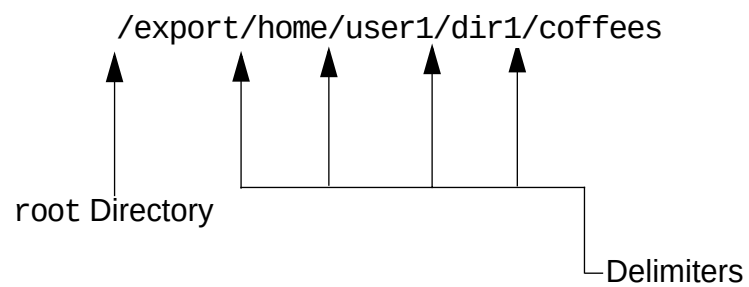


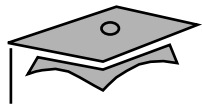
Pathnames

- Uniquely identify a particular file or directory by specifying its location in the directory tree.
- Similar to a road map that shows how to get from one place in the directory tree to another.
- The slashes (/) within a pathname are delimiters between object names.
- The first slash in a pathname always represents the root (/) directory.



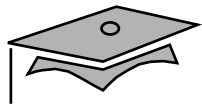
Pathnames





Absolute Pathname

- Absolute pathname – Specifies a file or directory in relation to the entire Solaris Operating Environment directory tree.
- Absolute pathnames always:
 - ▼ Start at the root (/) directory, then list each directory along the path to the final destination.
 - ▼ Use a slash (/) between each object name in the path.



Relative Pathname

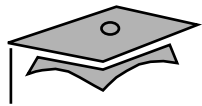
- Relative pathname – Describes the location of a directory or a file as it relates to the current directory
- Never begins with a slash (/) character
- Uses slashes (/) within the pathname as delimiters between object names



File Directory and Naming Conventions

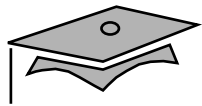
When creating files and directories, you must use the following conventions:

- Directory and file names can contain up to a maximum of 255 alphanumeric characters.
- Non-alphanumeric characters are allowed in file and directory names.
- Do not use special characters.
- Directory names do not contain extensions.



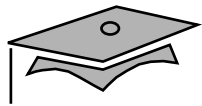
Changing Directories

- When you initially log in to the system, the current directory is set to your home directory. Change your current working directory at any time by using the `cd` command.
- Command format: `cd directory_name`



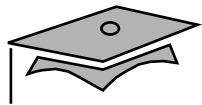
Displaying the Current Directory

- Use the `pwd` command to determine in which directory you are currently working.
- The `pwd` command displays the absolute pathname of the current working directory.
- Command format: `pwd`



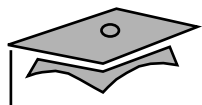
Displaying the Contents of a Directory

- Use the `ls` command to list the files and directories within the specified directory.
- Using the `ls` command with no argument simply displays the contents of the current directory.
- Command format:
`ls -option(s) pathname(s)`



Displaying a Long Listing

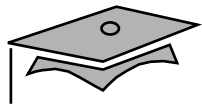
File type (- for regular file or d for directory, for example)	Permissions	Link count	Owner	Group	Size	Last modification date and time	File name
	drwxr-xr-x	5	user1	staff	512	Feb 22 14:51	dir1
	-rw-r--r--	1	user1	staff	0	Feb 22 14:51	file1



Metacharacters

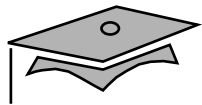
- Metacharacters are specific characters, generally symbols, used to represent a special meaning to the shell.
- Some examples of metacharacters include:

~ - + * ? [] ; \$ < > |



Exercise: Accessing Files and Directories

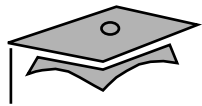
- Objectives
- Tasks
- Discussion
- Solutions



Check Your Progress

Check that you are able to accomplish or answer the following:

- Demonstrate the difference between absolute and relative pathnames
- Access files and directories within the directory tree using absolute and relative pathnames
- Use pathname abbreviations to access files and directories within the directory tree



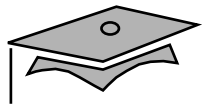
Check Your Progress

- List the contents of directories and determine file types
- Identify various shell metacharacters to abbreviate file names and pathnames



Module 4

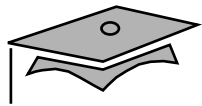
Directory and File Commands



Objectives

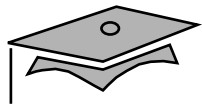
Upon completion of this module, you should be able to:

- Determine file types with the `file` command
- Display the contents of text files using the `cat`, `more`, `pg`, `head`, and `tail` commands
- Determine character, word, and line count using the `wc` command
- Create empty files or update modification times on existing files using the `touch` command



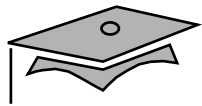
Objectives

- Use the `tee` command to create text within a file
- Create and remove directories using `mkdir` and `rmdir`
- Manage files and directories using the `mv`, `cp`, and `rm` commands
- Use commands to print a file, check print queue status, and cancel a print request
- Format and print the contents of files using the `pr` command



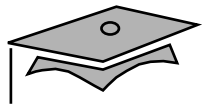
Determining File Type

- Use the `file` command to easily determine file type.
- Command format: `file filename(s)`
- Output from the `file` command is most often one of the following:
 - ▼ Text
 - ▼ Data
 - ▼ Executable or binary



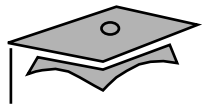
Browsing the Contents of a File

- To browse, or page, through the contents of a long text file, use the `more` command.
- Command format: `more filename(s)`



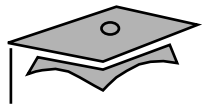
Displaying the Contents of a File

- The `cat` command, or concatenate command, displays the contents of a text file on the screen
- The `cat` command is also used to create short text files without having to use an editor
- Command format:
`cat filename(s)`
`cat > filename`



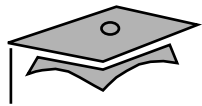
Viewing Long Files

- Use the `pg` command to view a file that is longer than one screen
- Command format: `pg filename(s)`



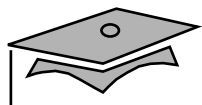
Displaying the First Few Lines of a File

- Use the `head` command to view the first few lines of one or more files. The first 10 lines are displayed by default if you omit the `-n` option.
- Command format: `head -n filename(s)`



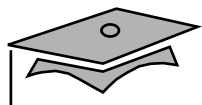
Displaying the Last Few Lines of a File

- Use the `tail` command to display the last few lines of one or more files.
 - ▼ Use the `-n` option to count relative to the end of the file.
 - ▼ Use the `+n` option to display the number of lines relative to the beginning of the file.
- Command format:
`tail -n filename(s)`
`tail +n filename(s)`



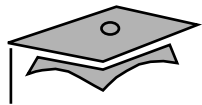
Displaying Lines, Words, and Characters in a File

- Use the `wc` command to display the number of lines, words, and characters contained in a file.
- Command format: `wc options filename(s)`



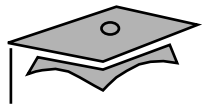
Creating Empty Files

- Use the touch command to create an empty file.
 - ▼ If the file name or directory name already exist, the touch command updates its modification and access time to the current date and time.
- Command format: touch *filename(s)*



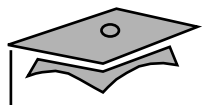
Creating and Appending Files Using the `tee` Command

- Use the `tee` command to create short text files.
 - ▼ If the file name does not exist, it is created and text is written into the file.
 - ▼ If the file name already exists, its contents are overwritten with the new text.
- Command format: `tee filename`



Creating Directories

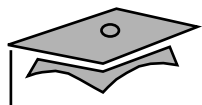
- Use the `mkdir` command to create directories. Directories can be created using absolute or relative pathnames.
- Gives you the ability to specify more than one directory name on the same command line, creating several directories at the same time.
- Command format:
`mkdir directory_name(s)`
`mkdir -p directory_pathname(s)`



Copying Files

- Use the `cp` command to copy files and directories.
- The `cp` command can copy the contents of a file to another file, or it can copy multiple file while preventing overwrites to existing files.
- Command format:

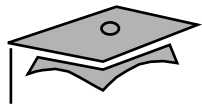
```
cp -i source_file destination_file  
cp -i source_file(s) destination_directory
```



Copying Directories

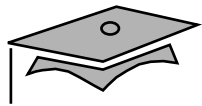
- Use the `cp -r` command to copy a directory and its contents to another directory.
- Command format:

```
cp -ri source_directory(s) destination_directory
```



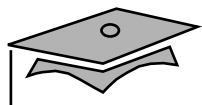
Moving and Renaming Files

- Use the `mv` command to move or rename a file or directory.
- Command format:
`mv -i source target_file`
`mv -i source target_directory`
- The `mv -i` option prompts for confirmation whenever the move would overwrite an existing target.
 - ▼ A yes response means the move should proceed.
 - ▼ A no response prevents `mv` from overwriting the target.



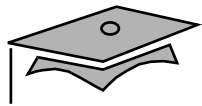
Removing Files

- Use the `rm` command to remove unwanted files and directories. Remove a single file or several files at once.
- Command format: `rm -i filename(s)`



Removing Directories

- Remove unwanted directories by using the `rmdir` and `rm` commands.
 - ▼ The `rmdir` command only deletes empty directories.
 - ▼ The `rm` command removes a directory that contains files.
- Command format:
`rmdir directory_name(s)`
`rm -ri directory_name(s)`



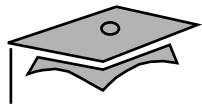
Command Line Printing

- The `lp` command queues text files for printing.
- From the command line, you can print:
 - ▼ ASCII text
 - ▼ PostScript™ files
- Command format: `lp -options filename(s)`



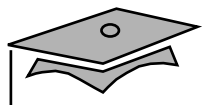
Displaying Printer Status and Queues

- The `lpstat` command displays the status of the printer queue.
- Command format: `lpstat -options`



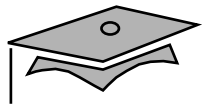
Removing a Print Request

- The `cancel` command enables you to cancel print requests previously sent with the `lp` command. To do this, first use the `lpstat` command to identify the *request-ID*.
- Command format: `cancel request-ID`



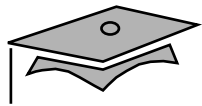
Format and Print a File

- Use the `pr` command to format and print the contents of a file according to different format options.
- Automatically prints the file contents to the terminal screen for viewing.
- Command format: `pr -options filename(s)`



Exercise: Using Directory and File Command

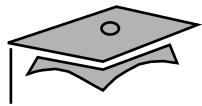
- Objectives
- Tasks
- Discussion
- Solutions



Check Your Progress

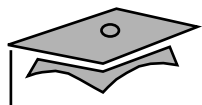
Check that you are able to accomplish or answer the following:

- Determine file types with the `file` command
- Display the contents of text files using the `cat`, `more`, `pg`, `head`, and `tail` commands
- Determine character, word, and line counts using the `wc` command
- Create empty files or update modification time on existing files using the `touch` command
- Use the `tee` command to create text within a file



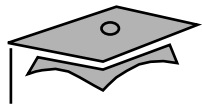
Check Your Progress

- Create and remove directories using the `mkdir` and `rmdir` commands
- Manage files and directories using `mv`, `cp`, and `rm`
- Use commands to print a file, check print queue status, and cancel a print request
- Format and print the contents of files using the `pr` command
- List the contents of directories and determine file types
- Identify various shell metacharacters to abbreviate file names and pathnames



Module 5

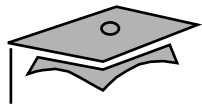
Searching for Files and Text



Objectives

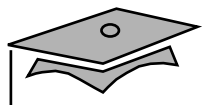
Upon completion of this module, you should be able to:

- Use the `find` command to locate files in the directory tree using specific search criteria
- Use the `cmp` and `diff` commands to compare the contents of text files for differences
- Sort the content of text files in alphabetical and numerical order using the `sort` command
- Search for regular expressions in the contents of one or more files using the commands `grep`, `egrep`, and `fgrep`



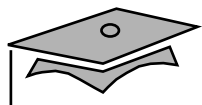
Locating Files Using the `find` Command

- Use the `find` command to locate a file in the directory tree:
 - ▼ Recursively descends directory tree in the pathname list, seeking those files which match the criteria.
 - ▼ As the files matching values are located, the path to each file displays on the screen.
- Command format:
`find pathname(s) expression(s) action(s)`



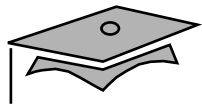
Arguments Used With the `find` Command

Argument	Definition
pathname	The directory path where the search originates.
expression	The search criteria specified by one or more options. Specifying multiple options causes <code>find</code> to treat the statement as an “and” request so all listed expressions must be verified as true.



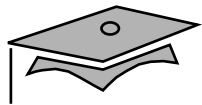
Expressions Used With the `find` Command

Expression	Definition
<code>-name <i>filename</i></code>	Finds files matching the specified <i>filename</i> . Metacharacters are acceptable if placed inside quotes.
<code>-size [+/-]<i>n</i></code>	Finds files that are larger than <i>+n</i> , or smaller than <i>-n</i> . The <i>n</i> represents 512-byte blocks.
<code>-atime [+/-]<i>n</i></code>	Finds files that have been accessed more than <i>+n</i> or less than <i>-n</i> days.
<code>-mtime [+/-]<i>n</i></code>	Finds files that have been modified more than <i>+n</i> or <i>-n</i> days.
<code>-user <i>loginID</i></code>	Finds all files that are owned by the <i>login ID</i> name.
<code>-type</code>	Finds a file type, for example: <code>f</code> (file) <code>d</code> (directory).
<code>-perm</code>	Finds files that have certain access permission bits.
<code>-print</code>	Instructs <code>find</code> to print the current pathname to the terminal screen. This is the default.



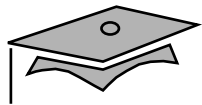
Actions Used With the `find` Command

Action	Definition
<code>-exec command {} \;</code>	Executes the specified <i>command</i> on each file located, automatically. The string <code>{}</code> <code>\;</code> must be included to instruct <code>find</code> to execute the <i>command</i> on every file identified by the search criteria. There must be a space before the backslash (<code>\</code>).
<code>-ok command {} \;</code>	Interactive form of <code>-exec</code> . Requires input before <code>find</code> applies the <i>command</i> on the file(s).



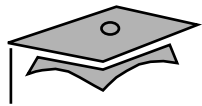
Locating Text Differences Using the `cmp` Command

- Use the `cmp` command to compare files to determine text differences.
- This command works with both binary and ASCII files.
- Command format: `cmp file1 file2`



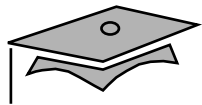
Locating Text Differences Using the `diff` Command

- The `diff` command is also used for finding differences between files.
- Results of `diff` command display line-by-line differences between two text files
- Command format: `diff option file1 file2`



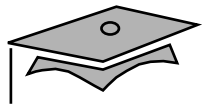
Sorting Data

- The `sort` command sorts text lines in one or more files and prints the results to the screen
- Quick and easy method to organize data in numeric or alphabetic order
- Command format: `sort options filename(s)`



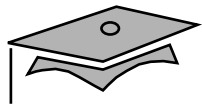
Searching for Text in Files Using the grep Command

- The grep command searches the contents of one or more files for a regular expression or character pattern.
- If found, grep prints every line containing the pattern to the screen and does not change the file content
- Command format:
`grep option(s) pattern filename(s)`



Searching for Text in Files Using the egrep Command

- The egrep command searches the contents of one or more files for a regular expression, using extended regular expression metacharacters.
- Uses all the same options and metacharacters as grep.
- Command format:
`egrep option(s) pattern filename(s)`



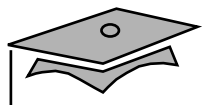
Searching for Text in Files Using the fgrep Command

- The fgrep command searches a file for a pattern.
- Differs from grep and egrep as it does not recognize metacharacters specified on the command line.
- Use fgrep to search for a specific pattern in a file that includes metacharacter symbols.
- Command format:
`fgrep option(s) pattern filename(s)`



Exercise: Locating Files and Text

- Objectives
- Tasks
- Discussion
- Solutions



Check Your Progress

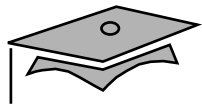
Check that you are able to accomplish or answer the following:

- Use the `find` command to locate files in the Solaris directory tree using specific search criteria
- Use the `cmp` and `diff` commands to compare the contents of text files for differences
- Sort the content of text files in alphabetical and numerical order using the `sort` command
- Search for regular expressions in the contents of one or more files using `grep`, `egrep`, and `fgrep`



Module 6

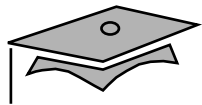
File Security



Objectives

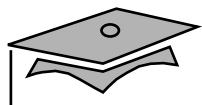
Upon completion of this module, you should be able to:

- Display file and directory permissions
- Define the standard permission types (read/write/execute)
- Use the `chmod` command to change permissions with symbolic mode or octal mode values
- Determine the default permissions assigned to newly created files and directories with `umask`



Security Overview

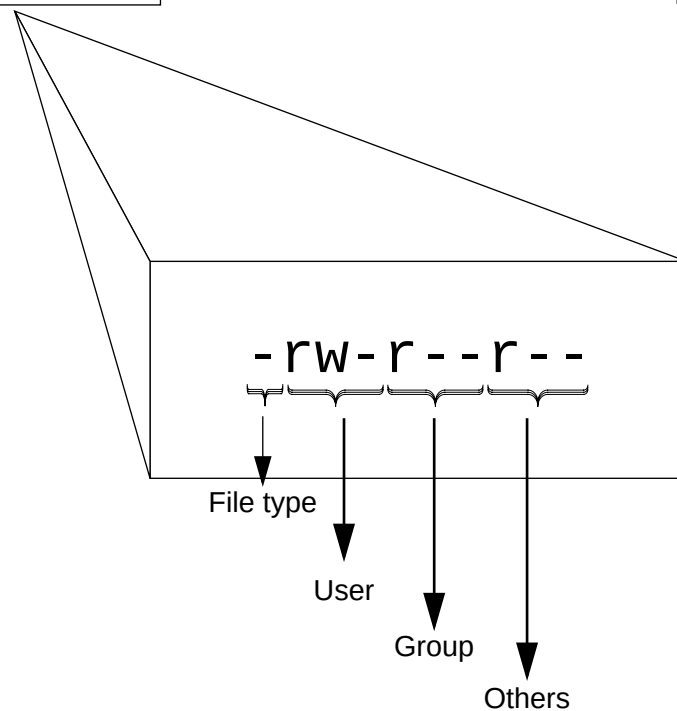
- The Solaris Operating Environment provides two basic measures to prevent unauthorized access to a system and to protect its data:
 - ▼ Authenticate a user's login by verifying the login ID and password existing in `/etc/passwd` and `/etc/shadow`
 - ▼ Automatically protect file and directory access by placing a standard set of access permissions when files and directories are created



Viewing File and Directory Permissions

View permissions on files and directories using `ls -l`.

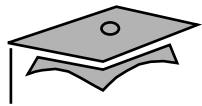
```
$ ls -l .profile  
-rw-r--r-- 1 user1 staff 560 Jun 11 11:23 .profile
```





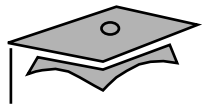
Permission Categories

- File type
- User (owner) permissions
- Group
- Others (public)



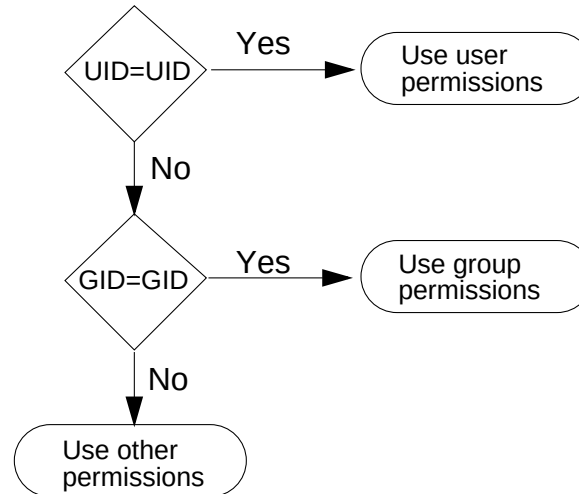
Determining Access to a File or Directory

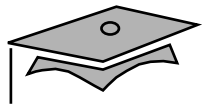
- Access to a file or a directory is determined by the user identification number (UID) and group identification number (GID)
 - ▼ UID – Identifies the user who created the directory or file and determines ownership
 - ▼ GID – Identifies the group of users who the directory or file belongs to
- To view these UID and GID numbers, use the `ls -n` command



Process For Determining Permissions

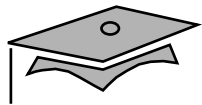
System process=file or directory





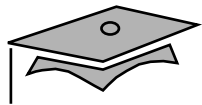
Types of Permissions

- File and directory access is protected by a standard set of default permissions, automatically assigned by Solaris Operating Environment when a file or directory is created
- Permissions control who can do what to a file or directory and are represented by the characters:
 - ▼ r (read)
 - ▼ w (write)
 - ▼ x (execute)
 - ▼ - (denied)



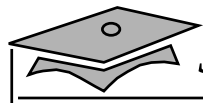
Types of Permissions

Permission	Permission Symbol	File	Directory
Read	r	File can be displayed or copied	Contents can be listed with the <code>ls</code> command
Write	w	File contents can be modified	Files can be added or deleted
Execute	x	File can be executed (shell scripts or executables only)	User can <code>cd</code> to the directory and issue the <code>ls -l</code> command on the directory

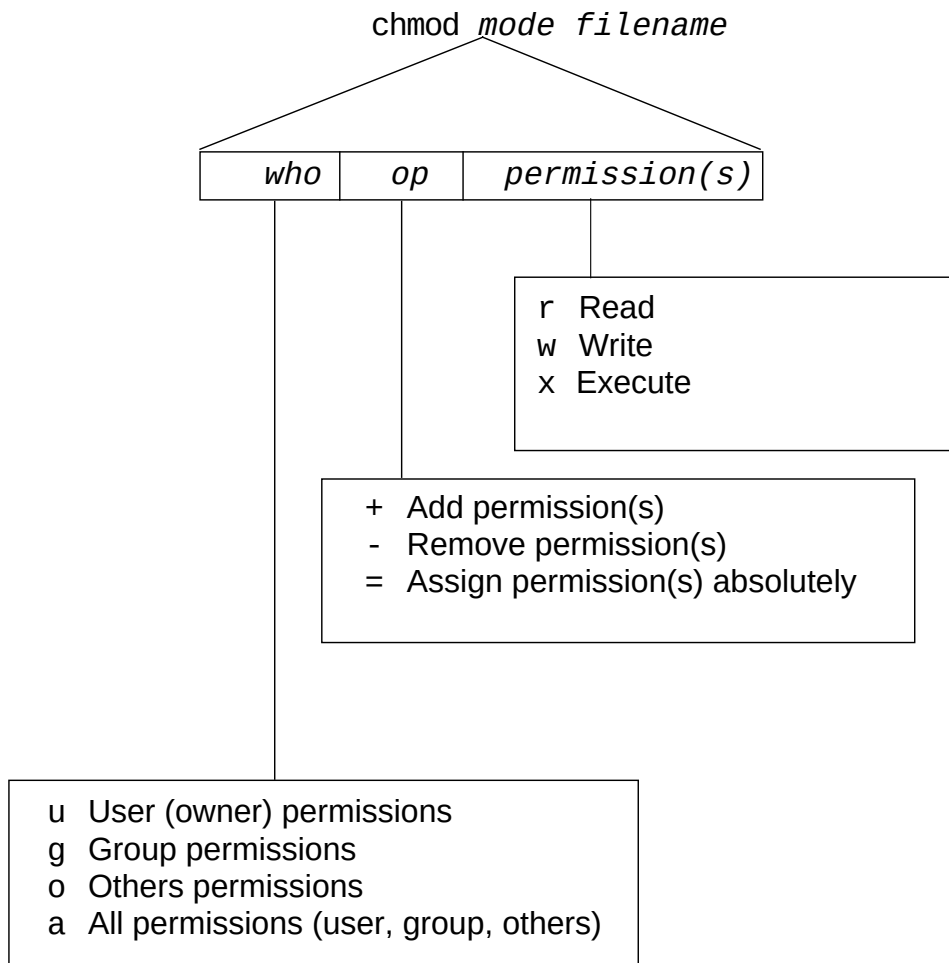


Changing Permissions

- Permissions set on newly created files or directories can be modified using the `chmod` command
- The `chmod` command can modify permissions specified in either *symbolic* mode or *octal* mode
 - ▼ Symbolic mode uses combinations of letters and symbols to add or remove permissions for each class of users
 - ▼ Octal mode uses numbers to represent each permission



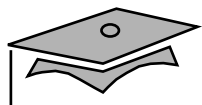
Changing Permissions





Permissions Assigned Octal Value

Octal Value	Permissions
4	Read
2	Write
1	Execute



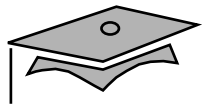
Octal Digits for Permission Sets

Octal Value	Permission Sets
7	r w x
6	r w -
5	r - x
4	r - -
3	- w x
2	- w -
1	- - x
0	- - -



Combined Values and Permissions

Octal Mode	Permissions
644	rw-r--r--
751	rxr-x--x
775	rxrwxr-x
777	rxrwxrwx



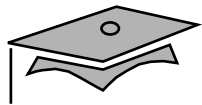
Default Permissions and the umask filter

- Determining the default permissions assigned to newly created files and directories is controlled by the umask filter.
- The umask filter is a three-digit octal value that refers to read/write/execute permissions for owner, group, and others
- The initial permission value specified by the system for a file creation is 666 (rw-rw-rw-)
- The initial permission value specified by the system for a directory creation is 777 (rwxrwxrwx)



Exercise: Changing File Permissions

- Objectives
- Tasks
- Discussion
- Solutions



Check Your Progress

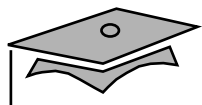
Check that you are able to accomplish or answer the following:

- Display file and directory permissions
- Define the standard permission types (read/write/execute)
- Use the `chmod` command to change permissions with symbolic mode or octal mode values
- Determine the default permissions assigned to newly created files and directories with `umask`



Module 7

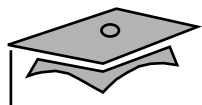
Visual (vi) Editor



Objectives

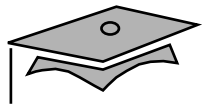
Upon completion of this module, you should be able to:

- Define the three modes of operation used by the `vi` editor
- Start the `vi` editor
- Position and move the cursor in `vi`
- Switch between `vi` modes
- Create and delete text



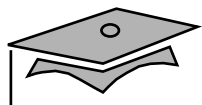
Objectives

- Copy or move text
- Set `vi` options
- Perform search and replace functions within `vi`
- Exit the `vi` editor



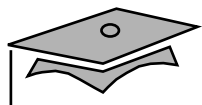
Introduction to vi

- Visual display (vi) editor is an interactive editor used to create or modify text files.
- All text editing with vi editor takes place in a buffer. Changes can either be written to disk or be discarded.
- For those who intend on becoming system administrators, you must know how to use vi.



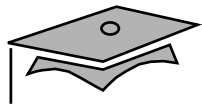
vi Modes

- The vi editor is a command-line editor that has three basic modes of operation:
 - ▼ Command mode
 - ▼ Edit mode
 - ▼ Last line mode



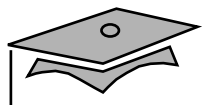
Command Mode

- The default mode for `vi`
- You can enter commands to delete, change, copy, and move text; position the cursor; search for text strings; or exit `vi`.



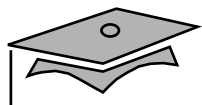
Edit Mode

- In this mode, you can enter text into a file.
- To instruct `vi` to enter edit mode, type one of the following three commands:
 - ▼ `i` (insert)
 - ▼ `o` (open)
 - ▼ `a` (append)



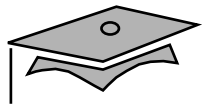
Last Line Mode

- While in command mode, you can do advanced editing commands by typing a colon (:), which places you at the bottom line of the screen.



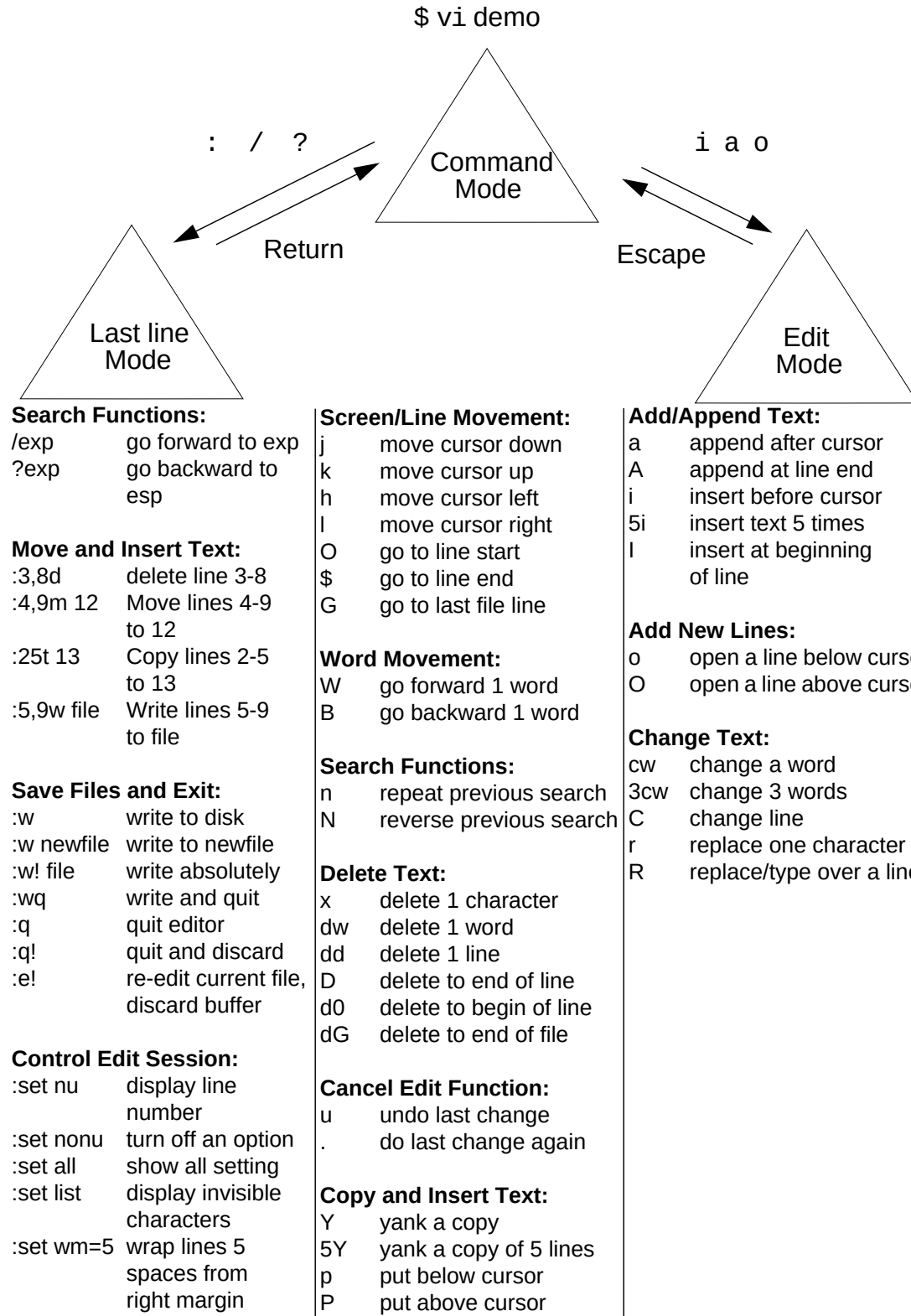
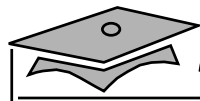
Invoking vi

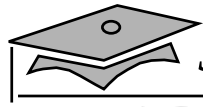
- To create a new file, invoke `vi` with a new file name by typing commands to create, edit, or view a file.
- Command format:
`vi option(s) filename view filename`



Customizing Your vi Session

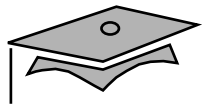
- vi editor includes options for customizing edit sessions, such as:
 - ▼ Displaying line numbers
 - ▼ Displaying invisible characters, such as tab and end-of-line characters
- The `set` command is used from command mode to control these options.





Exercise: Using the vi Editor

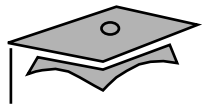
- Objectives
- Tasks
- Discussion
- Solutions



Check Your Progress

Check that you are able to accomplish or answer the following:

- Define the three modes of operation used by the `vi` editor
- Start the `vi` editor
- Position and move the cursor in `vi`
- Switch between `vi` modes
- Create and delete text



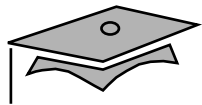
Check Your Progress

- Copy or move text
- Set `vi` options
- Perform search and replace functions within `vi`
- Exit the `vi` editor



Module 8

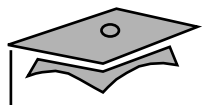
Archiving User Data



Objectives

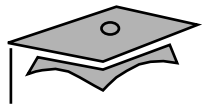
Upon completion of this module, you should be able to:

- Determine which commands are suitable for storing, viewing, or retrieving different types of files
- Demonstrate how to reduce the size of files and directories and store them to tape using the `compress` and `tar` commands
- Describe the steps for uncompressing or viewing a compressed file with the `uncompress` and `zcat` commands



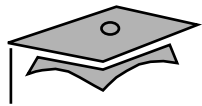
Objectives

- Use the `gzip` and `gunzip` commands to compress and uncompress files
- Use the `zip` command to package and compress multiple files and use `unzip` to uncompress a zipped archive file.
- Compress and copy multiple files to a single archive file in one step using the `jar` command
- Copy and extract files from an archive file or tape device with the `cpio` command



Objectives

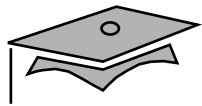
- Discuss the benefits of using volume management when working with removable media devices, such as disks and CD-ROMs
- Demonstrate how to access data on disks and CD-ROMs
- Use the `eject` command to remove disks and CD-ROMs from device drives



Overview of Archive Commands

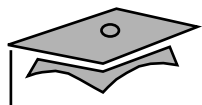
The commands available to easily store, locate, and retrieve files on a tape device or an archive file, include:

- `tar` – Creates and extracts files from a tape device or file archive
- `compress` and `uncompress` – Compresses and uncompresses a file
- `zcat` – Views a compressed file
- `gzcat` – Uncompresses and views a file at the same time



Overview of Archive Commands

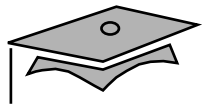
- `zip` and `unzip` – Packages and compresses files and uncompresses files
- `gzip` and `gunzip` – Compresses and uncompresses a file
- `jar` – Packages and compresses multiple files to a single archive file
- `cpio` – Copies and extracts files from a file archive or tape device



Archiving Files Using the tar Command

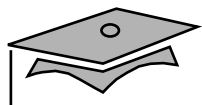
- The tar command archives and extracts files to and from a single file called a *tar file*. A tar file is usually a magnetic tape, but it can be any file.
- Command format:

```
tar function(s) archive_file/tape_device filename(s)
```



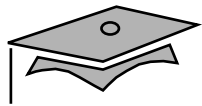
Compressing Files Using the `compress` Command

- The `compress` command is used to reduce the size of a file.
- When compressed, the named file is replaced by a new file with a `.Z` extension.
- Command format: `compress filename`



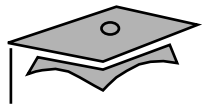
Uncompressing Files Using the uncompress Command

- The `uncompress` command restores a compressed file back to its original state.
- Command format: `uncompress filename -c`



Viewing Files Using the zcat Command

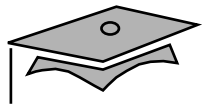
- The zcat command also allows you to view a file compressed with the compress command.
- The zcat command interprets the compressed data and displays the contents of the file as though it were not compressed.
- Command format: `zcat filename`



Compressing Files Using the jar Command

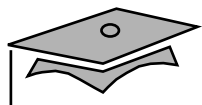
- The jar command is similar to the tar command, but compresses the named file(s) in the same step.
- This command copies and compresses multiple files into a single archive file.
- Command format:

```
jar options output_file filename(s)/directory(s)
```



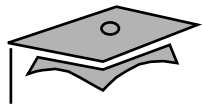
Using the zip Command

- The `zip` command is similar to the `jar` command in that it compresses named files. The `zip` command is a UNIX utility, while `jar` is a JavaTM application tool.
- When `zip` compresses a file, it adds the extension `.zip` to the file name.
- Command format: `zip filename(s) filename(s)`



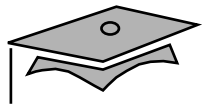
Using the gzip Command

- When gzip compresses a file, it adds the extension `.gz` to the file name.
- Command format:
`gzip filename filename filename`



The `cpio` Command

- The `cpio` (copy in/copy out) command archives or extracts file to a tape (or a single file).
- Some advantages of `cpio` are:
 - ▼ Packs data onto tape more efficiently than `tar`
 - ▼ Skips over bad spots in a tape when restoring files
 - ▼ Has ability to write files with different header formats for portability between system types
 - ▼ Creates multiple tape volumes
- Command format: `cpio options filename(s)`



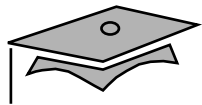
Detecting Removable Media Devices

- Volume management provides automatic detection of CD-ROMs.
- For floppy disks, volume manager needs to be informed every time a disk is inserted in the drive.
- To inform Volume Manager that a disk has been inserted in the drive the user invokes the `volcheck` command.
- Command format:
`volcheck options device_pathname`



Exercise: Saving and Restoring Files

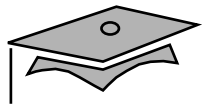
- Objectives
- Tasks
- Discussion
- Solutions



Check Your Progress

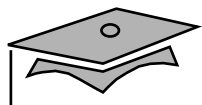
Check that you are able to accomplish or answer the following:

- Determine which commands are suitable for storing, viewing, or retrieving different types of files
- Demonstrate how to reduce the size of files and directories and store them to tape using the `compress` and `tar` commands
- Describe the steps for uncompressing or viewing a compressed file with the `uncompress` and `zcat` commands



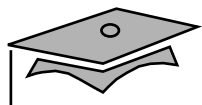
Check Your Progress

- Use the `gzip` and `gunzip` commands to compress and uncompress files
- Use the `zip` command to package and compress multiple files and use `unzip` to uncompress a zipped archive file.
- Compress and copy multiple files to a single archive file in one step using the `jar` command
- Copy and extract files from an archive file or tape device with the `cpio` command



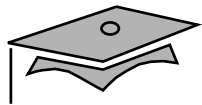
Check Your Progress

- Discuss the benefits of using volume management when working with removable media devices, such as disks and CD-ROMs
- Demonstrate how to access data on disks and CD-ROMs
- Use the `eject` command to remove disks and CD-ROMs from device drives



Module 9

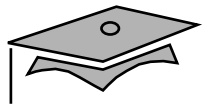
Remote Connections



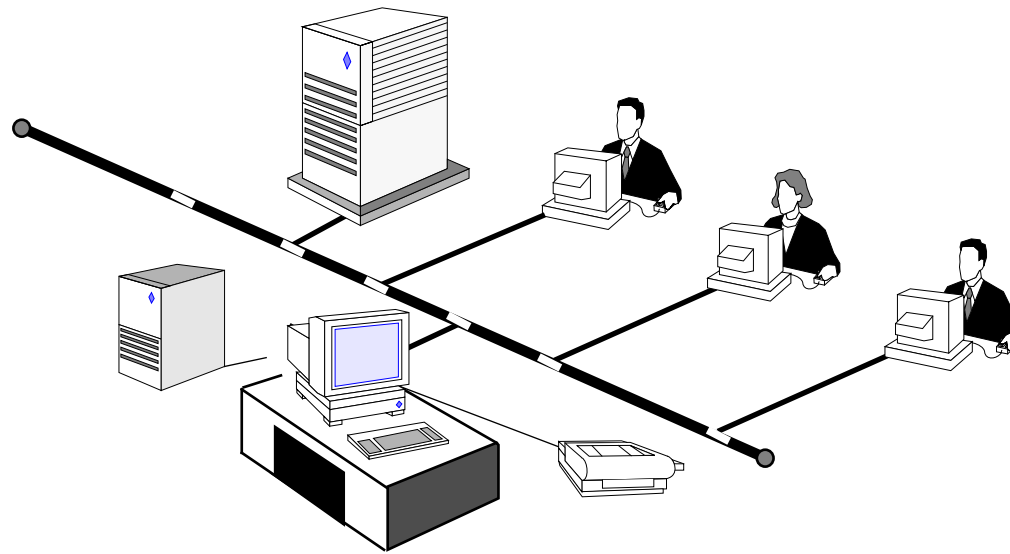
Objectives

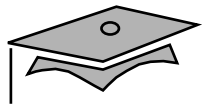
Upon completion of this module, you should be able to:

- Open a session on a remote machine using `telnet`
- Log in remotely to another machine on the network
- Use `ftp` to get a file from a remote system



Example Networking Environment





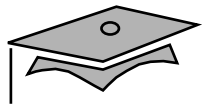
Network

- A connection that enables an exchange of information between machines. Two types of networks are:
 - ▼ Local area network (LAN) – A network that covers a small area, usually less than a few thousand feet or meters
 - ▼ Wide area network (WAN) – A network that can span thousands of miles or meters



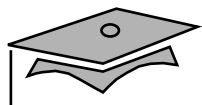
Host

- A computer system on a network.
 - ▼ The local host is the machine on which the user is currently working.
 - ▼ A remote host is a system accessed by a user from another system.



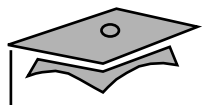
Using the telnet Command

- telnet is an application
- Uses Transmission Control Protocol/Internet Protocol (TCP/IP) to connect to another system
- Enables you to log in to a remote system and work in that environment.
- Command format: `telnet hostname`



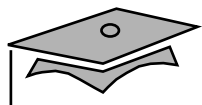
Using the `rlogin` Command

- Use the `rlogin` command to establish a remote login session on another workstation.
- Command format: `rlogin hostname -option`



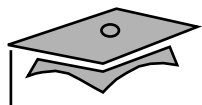
Specifying a Different Login ID

- Use the `-l` option to specify a different login ID for the remote login session.
- Command format: `rlogin username -l hostname`



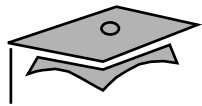
Using the rsh Command

- The rsh command is used to execute a program on a remote system.
- Command format:
`rsh -l username hostname command`
`rsh -l username IP_Address command`



Using the rcp Command

- The `rcp` command enables you to copy files or directories to and from another machine.
- Command format:
`rcp source_file hostname:destination_file`
`rcp hostname:source_file destination_file`



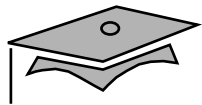
Using the ftp Command

- The ftp (File Transfer Protocol) command, part of an industry-standard application, is used to transfer files using ASCII or binary mode between systems using similar or dissimilar operating systems.
- Command format: `ftp hostname`



Exercise: Performing Network Basics

- Objectives
- Tasks
- Discussion
- Solutions



Check Your Progress

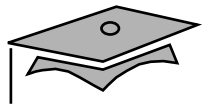
Check you are able to accomplish or answer the following:

- Open a session on a remote machine using `telnet`
- Login remotely to another machine on the network
- Use `ftp` to get a file from a remote system



Module 10

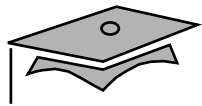
System Processes



Objectives

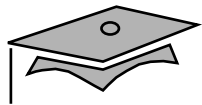
Upon completion of this module, you should be able to:

- Describe how to create a process
- View active processes on the system using the `ps` command
- Find a specific process using the `pgrep` command
- Discuss the purpose of signals for controlling process activity
- Terminate a process using `kill` and `pkill`
- Use job control commands to manage jobs running in the shell



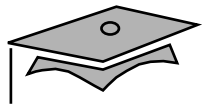
Process UID and GID

- For the kernel to know what a process is allowed to do, it must store information about who owns the process.
- The kernel stores two types of UIDs and GIDs for this purpose.



Viewing Processes and PIDs

- The `ps` (process status) command lists the processes currently running on the system.
- For each process, `ps` will display the process ID (PID), terminal identifier (TTY), cumulative execution time (TIME), and the command (CMD).
- Command format: `ps -options`

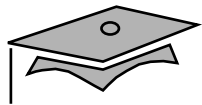


Displaying a Full Listing of All Processes

```
$ ps -ef | more
```

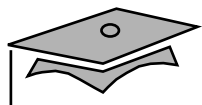
UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	0	0	0	16:46:41	?	0:01	sched
root	1	0	0	16:46:44	?	0:40	/etc/init -
root	2	0	0	16:46:44	?	0:00	pageout
root	3	0	0	16:46:44	?	4:33	fsflush
root	236	1	0	16:48:08	?	0:01	/usr/lib/saf/sac
root	844	1	0	12:12:10	?	0:00	/usr/lib/lpsched
aster	1292	1	0	06:48:51	console	0:01	-ksh
root	241	236	0	16:48:14	?	0:01	/usr/lib/saf/ttymon
rose	1400	321	0	20:03:11	?	0:00	/usr/openwin/bin/clock

--More--



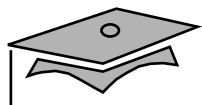
The pgrep Command

- The `pgrep` command gives you an efficient method for quickly finding a specific process by name.
- By default, `pgrep` displays the PID of any process that matches the specified criteria on the command line.
- Command format: `pgrep -options pattern`



The `kill` Command

- The `kill` command is used to send a signal to one or more running processes. This command is often used to terminate a process.
- Command format: `kill -signal process-ID`



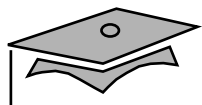
The `kill` Command

- The `kill` command is also used to terminate a process, by default using Signal 15. This command can terminate a process using its process name.
- Command format: `kill -options pattern`



Exercise: Manipulating System Processes

- Objectives
- Tasks
- Discussion
- Solutions



Check Your Progress

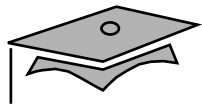
Check you are able to accomplish or answer the following:

- Describe how processes are created
- View active processes on the system using `ps`
- Find a specific process using the `pgrep` command
- Discuss the purpose of signals for controlling process activity
- Terminate a process using `kill` and `pkill`
- Use job control commands to manage jobs running in the shell



Module 11

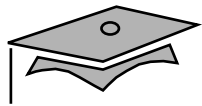
The Korn Shell



Objectives

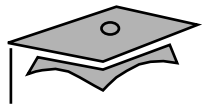
Upon completion of this module, you should be able to:

- Describe the functions of the Korn shell as a command interpreter
- Demonstrate the use of quoting to mask the special meaning of metacharacters by the Korn shell
- Define the terms standard input, standard output, and standard error
- Use metacharacters to redirect standard input, standard output and standard error



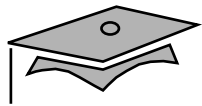
Objectives

- Connect two or more commands together using the pipe feature
- Implement the Korn shell file name completion mechanism
- Describe the two types of variables used by the Korn shell
- Use commands to view, set, and unset shell variables
- Invoke the `history` mechanism to repeat or edit previously executed commands



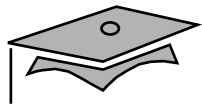
Objectives

- Use the `alias` utility to customize and abbreviate UNIX commands
- Create Korn shell functions to group UNIX commands into separate routines
- Define the Korn shell initialization files used to customize a user's environment

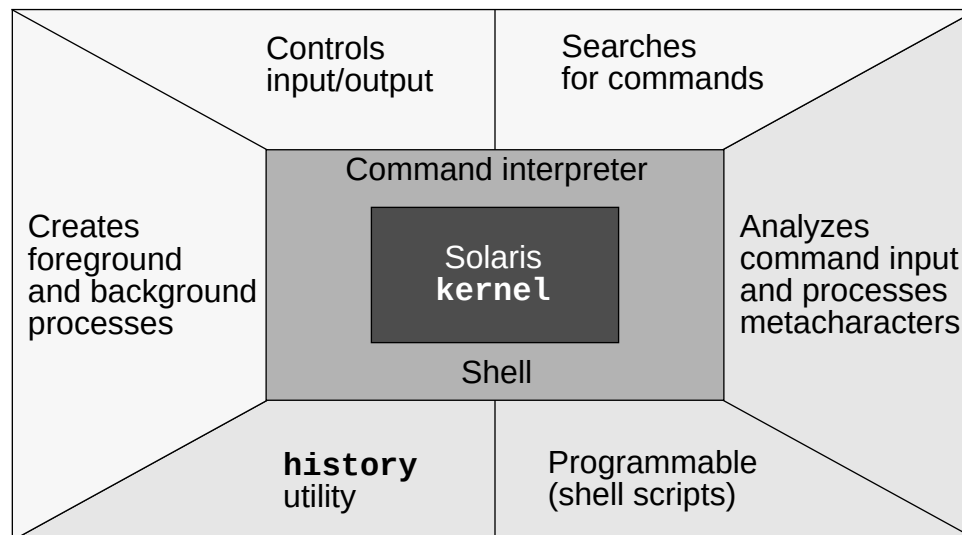


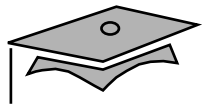
The Shell as a Command Interpreter

- The shell allows you to interact with the kernel by interpreting commands typed either on the command line or placed within a script file.
- The shell accepts, analyzes, and processes input from users or script files and also generates appropriate error messages.
- It is interchangeable, so you can choose which command interpreter to use at any time and change easily between the Bourne shell, the C shell, and the Korn shell.



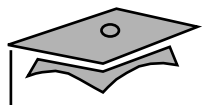
The Shell as a Command Interpreter



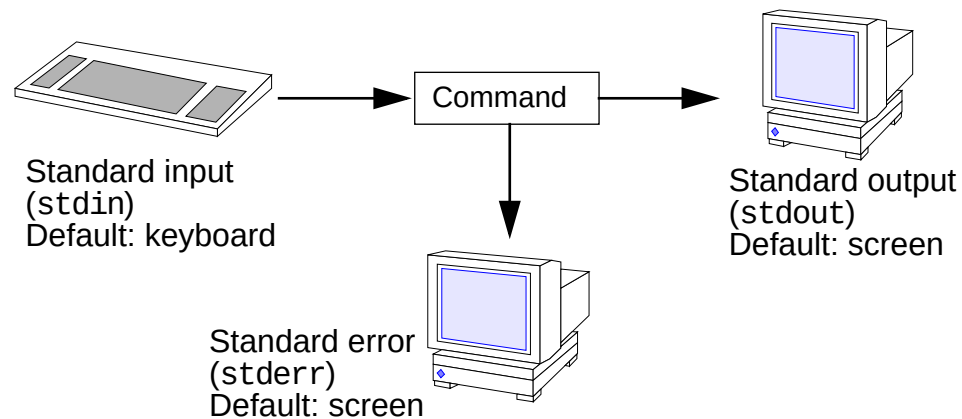


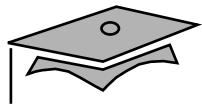
Redirecting Data I/O

- The shell typically receives, or reads, command input from the keyboard and displays, or writes, command output to the terminal screen.
- The user can instruct the shell to redirect command input and command output by using the redirect (< >) symbols.
- I/O redirection is commonly used for redirecting input and output to files on the command line or within shell scripts.



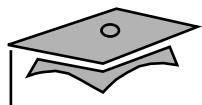
Redirecting Data I/O





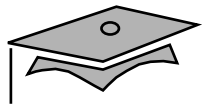
Redirecting stdin

- `command < filename` or `command 0< filename`
- The command reads its input from the file named *filename* instead of reading input from the keyboard.



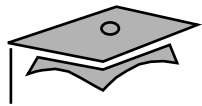
Redirecting stdout

- `command > filename` or `command 1> filename`
- The command output is directed to a file called *filename*.
 - ▼ If *filename* does not exist it is created.
 - ▼ If *filename* exists the redirection overwrites the contents of the file.



Redirecting stdout Using Append Mode

- `command >> filename`
- The command output is directed to a file called *filename* and appended to the end of the existing file content. If *filename* does not exist, it is created.



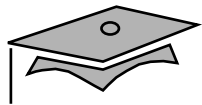
Redirecting stderr

- `command 2> /dev/null`
- Any command error messages are redirected to file `/dev/null`.
- This is useful to suppress error messages considered insignificant, so no error messages display on the terminal screen.



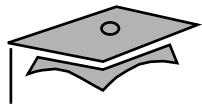
Redirecting stdout and stderr

- `command 1> filename 2>&1`
- The syntax `2>&1` instructs the shell to redirect stderr (2) to the same file that stdout (1) is directed to write its results.



The Pipe Feature

- The shell gives you the ability to effectively connect two commands together. This connection is known as a pipe (|).
- Building a pipe on the command line takes the output from one command and feeds it directly into the input of another command.
- A pipe is indicated by the character | and is placed between two commands.
- Command format: `command | command`

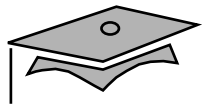


Options of the Korn Shell

- Options are switches which control the behavior of the Korn shell.
- To switch an option ON type:

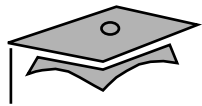
```
$ set -o option_name
```
- To switch an option OFF type:

```
$ set +o option_name
```



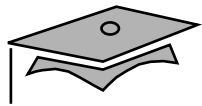
Variables of the Korn Shell

- A variable is a name that refers to a temporary storage area in memory. Variables contain information used for customizing the shell and information required by other processes so they function properly.
- Shell programming uses two types of variables:
 - ▼ Local variables — Private to the shell they were created in and not passed on to any processes started in that shell
 - ▼ Environment variables — Passed from parent process to child process, and so on



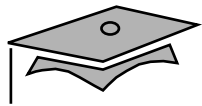
Customizing Korn Shell Variables

- The prompt variable PS1
 - ▼ The shell prompt string is stored in the shell variable PS1 and can be customized according to the preferences of the user.
- The PATH variable
 - ▼ Contains a list of directory path names, separated by colons.
 - ▼ When the user invokes a command from the command line, the shell searches these directories from left to right to locate the command to be executed.



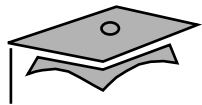
The History Mechanism

- The Korn shell keeps a history of recently entered commands.
- This history mechanism enables users to view, repeat, edit, or modify their previously executed commands.
- The `history` command displays the last 16 commands to standard output, by default.
- Command format: `history`



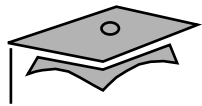
The `r` command

- The `r` command is a simple Korn shell command that allows users to repeat a command. For example:
- Command format: `r`



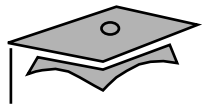
The Korn Shell Alias Utility

- An alias is a shorthand notation in the Korn shell to allow you to customize and abbreviate UNIX commands.
- An alias is defined by using the alias command.
- Command format: `alias name=command_string`



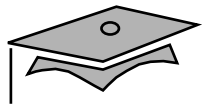
Korn Shell Functions

- Functions are a powerful feature of shell programming used to construct customized commands.
- A function is a group of UNIX commands organized as separate routines.
- Using a function involves two steps:
 - ▼ Defining the function
 - ▼ Invoking the function



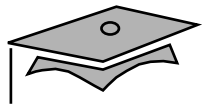
Defining a Function

- A function is defined using the general format:
 - ▼ Command format of:
`function name { command; . . . command; }`



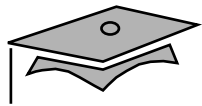
The ~/ .profile File

- The `.profile` is a user-defined initialization file that is executed once at login by the login shell and is found in the your home directory.
- It gives your the ability to customize and modify your working environment.



The ~/ .kshrc File

- This ENV file is often named `.kshrc`, by convention. It contains Korn shell variables and aliases.
- The ENV file is executed every time a Korn shell subshell is started.



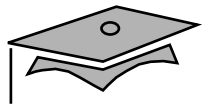
The .dtprofile file

- In the CDE environment, there is another initialization file called `.dtprofile`.
- It resides in the your home directory and determines generic and customized settings for CDE.
- The `.dtprofile` file means a your preferences for variable settings can overwrite default settings.



Exercise: Modifying the Korn Shell

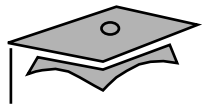
- Objectives
- Tasks
- Discussion
- Solutions



Check Your Progress

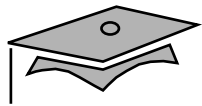
Check you are able to accomplish or answer the following:

- Describe the functions of the Korn shell as a command interpreter
- Demonstrate the use of quoting to mask the special meaning of metacharacters by the Korn shell
- Define the terms: standard input, standard output, and standard error
- Use metacharacters to redirect standard input, standard output and standard error



Check Your Progress

- Connect two or more commands together using the pipe feature
- Implement the Korn shell filename completion mechanism
- Describe the two types of variables used by the Korn shell
- Use commands to view, set, and unset shell variables
- Invoke the history mechanism to repeat or edit previously executed commands



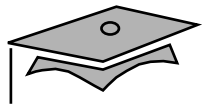
Check Your Progress

- Use the alias utility to customize and abbreviate UNIX commands
- Create Korn shell functions to group UNIX commands into separate routines
- Define the Korn shell initialization files used to customize a user's environment



Module 12

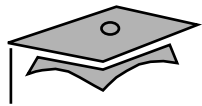
Introduction to sed and awk



Objectives

Upon completion of this module, you should be able to:

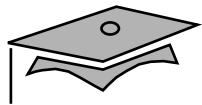
- Use the stream editor (`sed`) to edit the contents of a text file from the command line and send the results to standard output
- Issue `sed` commands to delete lines, print lines containing a pattern, add text to lines, or change characters using regular expression metacharacters
- Use `awk` to scan text files or standard input to display specific data, change data format, and add text to existing data



The Stream Editor

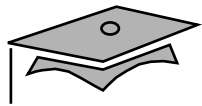
- The sed program is used to edit data in files without opening them in an interactive editor such as vi
- Allows you to specify edits, or modifications, to a file from the command line and send output to the screen
- Best used to make the same changes across multiple files quickly. It is also used to read scripts that administrators need.
- Command format:

```
sed [options] [address] command file(s) [>newfile]
```



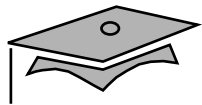
Text Processing Using the `awk` Command

- The `awk` command is a flexible text processor
- It scans a file line-by-line searching for lines that match a specified pattern and perform selected actions on those lines.
- Some basic uses of `awk` include making format changes to data, reordering columns, and adding to existing text.
- Command format: `awk '{ action }' [ filename]`



Exercise: Using sed and awk

- Objectives
- Tasks
- Discussion
- Solutions



Check Your Progress

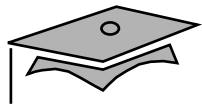
Check you are able to accomplish or answer the following:

- Use the stream editor (`sed`) to edit the contents of a text file from the command line and send the results to standard output
- Issue `sed` commands to delete lines, print lines containing a pattern, add text to lines, or change characters using regular expression metacharacters
- Use `awk` to scan text files or standard input to display specific data, change data format, and add text to existing data



Module 13

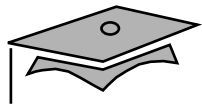
Reading Shell Scripts



Objectives

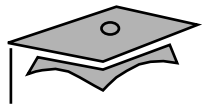
Upon completion of this module, you should be able to:

- Identify which shell program interpreting the lines of a script
- Explain how command line arguments are passed to a shell script with special variables called positional parameters
- Demonstrate the use of three conditional commands: `if`, `test`, and `case`
- Interpret the contents of a simple Bourne shell administration script



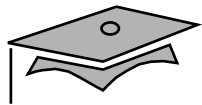
The Basics of Shell Scripts

- A shell script is simply an ASCII file that contains a sequence of commands and comments.
- Comments are text used to document what the script does and describes what the lines within the script are supposed to do when it is executed.
- Comments are preceded by a hash symbol (#) and can be on a line by themselves or on the same line following a command.



Determining the Type of Shell Script

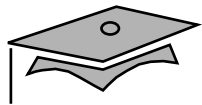
- The top line of a script identifies the shell program that will execute the lines in the script.
- For example, for a Bourne shell script the first line would read: `#!/bin/sh`
- The `#!` is used by the kernel to identify the program that will interpret the lines in the script.



Creating a Basic Shell Script

1. With the `vi` editor, create a file called `my.script`, enter each command followed by a Tab and hash (`#`) symbol to define it as a comment.
2. Make the file executable, and run the new script by typing its name on the command line.

```
$ vi my.script
who # To view who is logged on the system.
date # To view the current date and time
ls -l # To view files in current directory
:wq
$ chmod 755 my.script
$ ./my.script
```



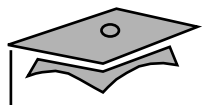
Bourne Shell Programming

- The standard administration scripts used to manage the Solaris Operating Environment are Bourne shell scripts.
- To successfully control or modify the behavior of the operating environment, system administrators must be able to read, modify, and customize the contents of these various shell script files.



Bourne Shell Scripts

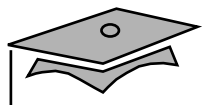
- To be able to read and comprehend the contents of a basic shell script, you must be able to:
 - ▼ Understand how arguments are passed to a script, through the use of special built-in variables called positional parameters.
 - ▼ Identify and analyze simple conditional constructs and flow control.



Positional Parameters

- Special shell built-in variables, called *positional parameters*, are used to pass arguments from the command line into a script.
- On the command line, each word, separated by a space, that follows the script name is called an argument. These arguments are referenced in the shell script with positional parameters.
- Command format:

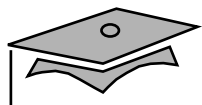
scriptname argument1 argument2 argument3 ...



Conditional Commands

- Conditional commands allow you to perform some tasks based on whether or not a condition succeeds or fails.
- The simplest form of a conditional is the `if` command, which enables the user to test a condition and then change the flow of a script's execution based on the results of the test.
- Command format:

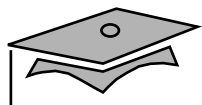
```
if  command
then
    command
    command
fi
```



The test Command

- The built-in command `test` is often used as the command following the `if` command.
- The `test` command evaluates an expression, and if the result is true, it returns an exit status of zero. Otherwise the result is false, and it returns a non-zero exit status.
- Command format:

```
if test expression
then
    command
fi
```



The case Command

- Used when there are many conditions to test.
- Command format:

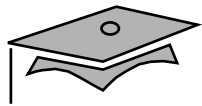
```
case value in
value1 )      command
               command ;;

value2 )      command
               command ;;

value3 )      command
               command ;;

* )           command ;;

esac
```



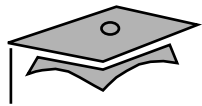
The `exit` Command

- The built-in shell command called `exit` gives the user the ability to immediately terminate execution of a shell script.
- The `exit` command is frequently used as a convenient method to end or breakout of a script and return to the command line.
- Command format: `exit n`



Exercise: Introduction to Scripting

- Objectives
- Tasks
- Discussion
- Solutions



Check Your Progress

Check you are able to accomplish or answer the following:

- Identify which shell program will be interpreting the lines of a script
- Explain how command line arguments are passed to a shell script with special variables called positional parameters
- Demonstrate the use of three conditional commands: `if`, `test`, and `case`
- Interpret the contents of a simple Bourne shell administration script

Copyright 2000 Sun Microsystems Inc., 901 San Antonio Road, Palo Alto, California 94303, Etats-Unis. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a.

Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun. Des parties de ce produit pourront être dérivées du système Berkeley 4.3 BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company Ltd.

Sun, Sun Microsystems, le logo Sun, Solaris Operating Environment, Common Desktop Environment (CDE), admintool, SPARC, et SunOS sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays.

Toutes les marques SPARC sont utilisées sous licence sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays.

Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

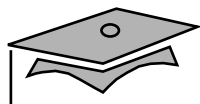
L'accord du gouvernement américain est requis avant l'exportation du produit.

Le système X Window est un produit de X Consortium, Inc.

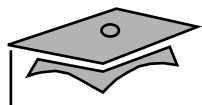
LA DOCUMENTATION EST FOURNIE "EN L'ETAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFAÇON.

Seminar Contents

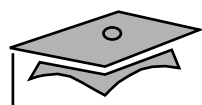
About This Course	3
Course Goal	4
Course Overview	5
Course Map	6
Module-by-Module Overview	7
Course Objectives	9
Skills Gained by Module	11
Guidelines for Module Pacing	12
Topics Not Covered	13
How Prepared Are You?	14
Introductions	15
How to Use the Course Materials	16
How to Use the Icons	17
Typographical Conventions and Symbols	19
 Introduction to the Solaris 8	
Operating Environment	1-1
Objectives	1-2
Main Hardware Components of a Computer	1-3
The Solaris Operating Environment	1-5
The SunOS Operating System	1-6
The kernel	1-7
The Shell	1-9
The Bourne Shell	1-11
The C Shell	1-12
The Korn Shell	1-13



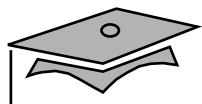
Exercise: Using the Solaris 8 Operating Environment	1-14
Check Your Progress	1-15
 Accessing the System	2-1
Objectives	2-2
User Accounts	2-4
Logging In Using the Command Line	2-7
Password Requirements	2-8
Changing Your Password	2-9
Securing Your CDE Session	2-10
Basic UNIX Commands	2-11
Control Characters	2-13
Viewing Online Documentation	2-14
Determining Current Users	2-15
Identifying the User Name	2-16
Identifying User Group Details	2-17
Entering Multiple Commands from a Single Command Line	2-18
Exercise: Accessing the System	2-19
Check Your Progress	2-20
 Accessing Files and Directories	3-1
Objectives	3-2
The Directory Tree	3-3
Absolute Pathname	3-6
Relative Pathname	3-7
File Directory and Naming Conventions	3-8
Changing Directories	3-9
Displaying the Current Directory	3-10
Displaying the Contents of a Directory	3-11
Displaying a Long Listing	3-12



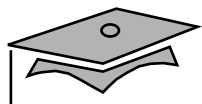
Metacharacters	3-13
Exercise: Accessing Files and Directories	3-14
Check Your Progress	3-15
 Directory and File Commands	4-1
Objectives	4-2
Determining File Type	4-4
Viewing Long Files	4-7
Displaying the First Few Lines of a File	4-8
Displaying the Last Few Lines of a File	4-9
Displaying Lines, Words, and Characters in a File	4-10
Creating Empty Files	4-11
Creating and Appending Files Using the tee Command	4-12
Creating Directories	4-13
Copying Files	4-14
Copying Directories	4-15
Moving and Renaming Files	4-16
Removing Files	4-17
Removing Directories	4-18
Command Line Printing	4-19
Displaying Printer Status and Queues	4-20
Removing a Print Request	4-21
Format and Print a File	4-22
Exercise: Using Directory and File Command	4-23
Check Your Progress	4-24
 Searching for Files and Text	5-1
Objectives	5-2
Locating Files Using the find Command	5-3
Arguments Used With the find Command	5-4



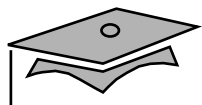
Expressions Used With the <code>find</code> Command	5-5
Actions Used With the <code>find</code> Command	5-6
Locating Text Differences Using the <code>cmp</code> Command	5-7
Locating Text Differences Using the <code>diff</code> Command	5-8
Sorting Data	5-9
Searching for Text in Files Using the <code>grep</code> Command	5-10
Searching for Text in Files Using the <code>egrep</code> Command	5-11
Searching for Text in Files Using the <code>fgrep</code> Command	5-12
Exercise: Locating Files and Text	5-13
Check Your Progress	5-14
 File Security	6-1
Objectives	6-2
Security Overview	6-3
Viewing File and Directory Permissions	6-4
Permission Categories	6-5
Determining Access to a File or Directory	6-6
Process For Determining Permissions	6-7
Types of Permissions	6-8
Changing Permissions	6-10
Permissions Assigned Octal Value	6-12
Octal Digits for Permission Sets	6-13
Combined Values and Permissions	6-14
Default Permissions and the <code>umask</code> filter	6-15
Exercise: Changing File Permissions	6-16
Check Your Progress	6-17
 Visual (vi) Editor	7-1
Objectives	7-2
Introduction to <code>vi</code>	7-4



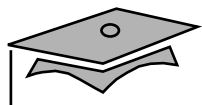
vi Modes	7-5
Command Mode	7-6
Edit Mode	7-7
Last Line Mode	7-8
Invoking vi	7-9
Customizing Your vi Session	7-10
Exercise: Using the vi Editor	7-12
Check Your Progress	7-14
 Archiving User Data	8-1
Objectives	8-2
Overview of Archive Commands	8-5
Archiving Files Using the tar Command	8-7
Compressing Files Using the compress Command	8-8
Uncompressing Files Using the uncompress Command	8-9
Viewing Files Using the zcat Command	8-10
Compressing Files Using the jar Command	8-11
Using the zip Command	8-12
Using the gzip Command	8-13
The cpio Command	8-14
Detecting Removable Media Devices	8-15
Exercise: Saving and Restoring Files	8-16
 Remote Connections	9-1
Objectives	9-2
Example Networking Environment	9-3
Network	9-4
Host	9-5
Using the telnet Command	9-6
Using the rlogin Command	9-7



Specifying a Different Login ID	9-8
Using the rsh Command	9-9
Using the rcp Command	9-10
Using the ftp Command	9-11
Exercise: Performing Network Basics	9-12
 System Processes	10-1
Objectives	10-2
Process UID and GID	10-3
Viewing Processes and PIDs	10-4
Displaying a Full Listing of All Processes	10-5
The pgrep Command	10-6
The kill Command	10-7
The pkill Command	10-8
Exercise: Manipulating System Processes	10-9
 The Korn Shell	11-1
Objectives	11-2
The Shell as a Command Interpreter	11-5
The Shell as a Command Interpreter	11-6
Redirecting Data I/O	11-7
Redirecting stdin	11-9
Redirecting stdout	11-10
Redirecting stdout Using Append Mode	11-11
Redirecting stderr	11-12
Redirecting stdout and stderr	11-13
The Pipe Feature	11-14
Options of the Korn Shell	11-15
Variables of the Korn Shell	11-16
Customizing Korn Shell Variables	11-17



The History Mechanism	11-18
The <code>r</code> command	11-19
The Korn Shell Alias Utility	11-20
Korn Shell Functions	11-21
Defining a Function	11-22
The <code>~/.profile</code> File	11-23
The <code>~/.kshrc</code> File	11-24
The <code>.dtprofile</code> file	11-25
Exercise: Modifying the Korn Shell	11-26
Check Your Progress	11-27
 <i>Introduction to sed and awk</i>	<i>12-1</i>
Objectives	12-2
The Stream Editor	12-3
Text Processing Using the <code>awk</code> Command	12-4
Exercise: Using <code>sed</code> and <code>awk</code>	12-5
Check Your Progress	12-6
 <i>Reading Shell Scripts</i>	<i>13-1</i>
Objectives	13-2
The Basics of Shell Scripts	13-3
Determining the Type of Shell Script	13-4
Creating a Basic Shell Script	13-5
Bourne Shell Programming	13-6
Bourne Shell Scripts	13-7
Positional Parameters	13-8
Conditional Commands	13-9
The <code>test</code> Command	13-10
The <code>case</code> Command	13-11
The <code>exit</code> Command	13-12



Exercise: Introduction to Scripting	13-13
Check Your Progress	13-14