

Fundamentals UNIX Shell Programming

Fundamentals

Chapter 1:

Objectives

You will learn:

- the command process.
- running complex commands.
- interactive processing of commands.
- using files and directories.
- scripting.
- working with files.
- working with utilities.

Fundamentals

UNIX Shell Programming

Shell Basics

- In UNIX, commands and utilities are used for developing scripts and programmatic code. There are core commands, such as ls and cd, and more sophisticated tools such as awk, sed, and the shell.
- The core commands are straightforward.
- The tools will take longer to master, but will provide significant improvements in productivity.

SYS-ED/Computer Education Techniques, Inc.

1: 3

Command: What is it

- In UNIX, a command is a program that can be executed.
- In order to run a command in UNIX, its name is typed at the prompt and the Enter key invoked.
\$ date [ENTER]
- The date command has been entered. This command displays the current day, date, time, and year. After the current date appears, the \$ character is displayed.
- A second example of running a command is:
\$ who
- This command displays a list of all the people, or users, currently using the UNIX machine.

SYS-ED/Computer Education Techniques, Inc.

1: 4

Fundamentals

UNIX Shell Programming

Simple Commands

- A command is executed by providing the name at the prompt:
\$ command
 - The command is the name of the command to be executed.
- Commands in UNIX can be as basic as who and date, or more complex commands with a large variety of options.

SYS-ED/Computer Education Techniques, Inc.

1: 5

Complex Commands

- The who command is used for gathering information about the logged on user – yourself.
 - \$ who am i
- The who command is a complex command, it consists of a command name and a list of arguments.
- The arguments am and i change the behavior of the who command and the information to listed. In UNIX, most commands accept arguments that modify their behavior.
- The formal syntax for a complex command is:
\$ command argument1 argument2 argument3 ...
argumentN

SYS-ED/Computer Education Techniques, Inc.

1: 6

Fundamentals

UNIX Shell Programming

Compound Commands

- UNIX has the capability to combine simple and complex commands together to obtain compound commands.
- A compound command consists of a list of simple and complex commands separated by the semicolon character (;).
 - An example of a complex command is:
\$ date ; who am i ;
- The compound command consists of the simple command date and the complex command who am i.
- The formal syntax for a complex command is:
\$ *command1* ; *command2* ; *command3* ; ... ; *commandN* ;

SYS-ED/Computer Education Techniques, Inc.

1: 7

Command Separators

- The semicolon character (;) is treated as a command separator, which indicates where one command ends and another begins.
- Individual simple and complex commands can be terminated using the semicolon character.
 - For example, the commands:
\$ date And \$ date ;
produce the same output due to the order in which the commands execute.
- In the first case, the simple command date executes, and the prompt returns.
- In the second case, from the perspective of the UNIX system, a complex command is executing.
 - It begins by executing the first command in the complex command.
 - In this case, it is the date command.
- When this command finishes, the UNIX system attempts to execute the next command.
 - Since there are no other commands to execute, the prompt returns.

SYS-ED/Computer Education Techniques, Inc.

1: 8

Fundamentals

UNIX Shell Programming

Shell: What is it

- The shell provides an interface to the UNIX system.
- It gathers input from and executes programs based on that input.
 - When a program finishes executing, it displays that program's output.
- It is for this reason, that the shell is referred to as the UNIX system's command interpreter.
 - The UNIX shell is similar to the DOS shell, COMMAND.COM.
- A significant benefit to the UNIX shell is the fact that it is much more than a command interpreter. It is also a programming language, with conditional statements, loops, and functions.

SYS-ED/Computer Education Techniques, Inc.

1: 9

Shell Prompt

- When the prompt is displayed, a command can be typed. The shell reads input after Enter from the keyboard has been invoked. It determines the command to be executed by looking at the first word of the input.
 - A word is an unbroken set of characters.
 - Spaces and tabs separate words.
- To the shell, input will appear as:
\$ word1 word2 word3 ... wordN
- The shell always picks word1 as the name of the command to be executed.
 - If there is only a single word such as:
 - \$ date
 - the shell's job is straightforward; it executes the command.

SYS-ED/Computer Education Techniques, Inc.

1: 10

Fundamentals

UNIX Shell Programming

Shell Prompt

- If there are more words such as:
 - \$ who am i
- The shell passes the extra words as arguments to the command specified by word1.

SYS-ED/Computer Education Techniques, Inc.

1: 11

Shells: Different Types

- The prompt that is displayed is dependent upon on the type of shell that is being used.
- The prompt will be different from the \$ prompt that is being used.
- In UNIX there are two major types of shells:
 - The Bourne shell including sh, ksh, and bash.
 - When a Bourne-type shell is being used, the default prompt is the \$ character.
 - The C shell including csh and tcsh.
 - When a C-type shell is being used, the default prompt is the % character.

SYS-ED/Computer Education Techniques, Inc.

1: 12

Fundamentals UNIX Shell Programming

Scripting

UNIX System

- Utilities are programs which can be executed.
 - who and date are examples of utilities.
- Commands are somewhat different than utilities.
 - The term utility refers to the name of a program.
 - The term command refers to the program and any arguments that are specified which to change its behavior.
 - The term command can be used interchangeably with utility when referring to a simple command; when only the program name to execute is given.
- The kernel is the core of the UNIX system.
 - It provides utilities with a means of accessing a machine's hardware.
 - It also handles the scheduling and execution of commands.
- When a machine is turned off, both the kernel and the utilities are stored on the machine's hard disks.
 - When the computer is booted, the kernel is loaded from disk into memory.
 - The kernel remains in memory until the machine is turned off.
- Utilities, on the other hand, are stored on disk and loaded into memory only when they are executed.

Fundamentals

UNIX Shell Programming

Logging In

- When connecting to a UNIX system, there typically will be a prompt:
 - login:
- Username will need to be entered at this prompt.
- After username is entered, another prompt is presented:
login: ranga
Password:
- It will be necessary to enter a password at this prompt.
- These two prompts are presented by the `getty` program.
- These are the tasks:
 1. Display the prompt login.
 2. Wait for a user to type a username.
 3. After a username has been entered, display the password prompt.
 4. Wait for a user to enter a password.
 5. Give the username and password entered by the user to the login command and exit.
- After login receives the username and password, it looks through the file `/etc/passwd` for an entry matching the information that has been provided.
- If it finds a match, login executes a shell and exits.
- The shell that login executes is specified in the file `/etc/passwd`.

SYS-ED/Computer Education Techniques, Inc.

1: 15

Shell Initialization

- When the login program executes a shell, that shell is uninitialized.
- An uninitialized shell, does not have the parameters required to function correctly as being properly defined.
- The shell undergoes a phase called initialization to set up these parameters.
- This is typically a two step process that involves the shell reading the following files:
 - `/etc/profile`
 - `profile`
- The process is:
 1. The shell checks to see whether the file `/etc/profile` exists.
 2. If it exists, the shell reads it. Otherwise, this file is skipped. No error message is displayed.
 3. The shell checks to see whether the file `.profile` exists in the home directory.
 - The home directory is the directory where the user starts after logging in.
 4. If the home directory exists, the shell reads it; otherwise, the shell skips it.
 - No error message is displayed.As soon as both of these files have been read, the shell displays a prompt:
\$

SYS-ED/Computer Education Techniques, Inc.

1: 16

Fundamentals

UNIX Shell Programming

Interactive Versus Noninteractive Shells

- When the shell displays a prompt to the user, it is running in interactive mode.
- With interactive mode, the shell expects to receive input from the user and execute the commands that have been specified.
- This typically is the mode of the shell that most users will be working are familiar with:
 - Log in, execute some commands, and log out.
 - Upon logging out by running the exit command, the shell exits.
- The shell can also be run in non interactive mode.
 - In this mode, the shell reads commands stored in a file and executes.
 - When it reaches the end of the file, the shell exits.

SYS-ED/Computer Education Techniques, Inc.

1: 17

How login Starts a Shell

- When the login program starts a shell, it executes the following command:
`/bin/sh`
- By issuing this command, it puts the shell into interactive mode. A shell can be started in interactive mode by issuing the same command at the prompt:
`$ /bin/sh`
`$`
 - The first prompt `$` is displayed by the shell that login started;
 - The second prompt is displayed by the shell that has been started.The exit command is used for exiting from this shell:
`$ exit`

SYS-ED/Computer Education Techniques, Inc.

1: 18

Fundamentals

UNIX Shell Programming

How to Start the Shell Noninteractively

- A shell script is a list of commands stored in a file that the shell executes noninteractively.
- The shell executes noninteractively as follows:
 - \$ /bin/sh filename
 - The filename is the name of a file that contains commands to execute.
 - For example, the compound command:
\$ date ; who
 - Can be placed into a file called logins.
 - A file called logins is opened in an editor and the command typed in.
 - Assuming that the file is located in the current directory, after the file is saved, the command can run as:
\$ /bin/sh logins
 - This executes the compound command and displays its output.

SYS-ED/Computer Education Techniques, Inc.

1: 19

Initialization File Contents

- Typically, the shell initialization files are quite short. They are designed to provide a complete working environment with minimal overhead for both interactive and noninteractive shells.
- The file /etc/profile is maintained by the UNIX system administrator and contains shell initialization information required by all users on a system.
- Shell customization information can be added to the “.profile file”.
- The minimum set of information that needs to be configured includes:
 - The terminal type.
 - List of directories in which to locate commands.
 - List of directories in which to locate manual pages for commands.

SYS-ED/Computer Education Techniques, Inc.

1: 20

Fundamentals

UNIX Shell Programming

Setting the Terminal Type

- The terminal type is automatically configured by either the login or getty programs. Sometimes, the autoconfiguration process will guess the terminal incorrectly. This can occur when using a dial-up or modem connection.
- If a terminal is set incorrectly, the output of commands will be irregular or it may not be possible to interact with the shell properly.
- In order to make certain, that this is not the case, most users set their terminal to a minimal common denominator:
TERM=vt100

SYS-ED/Computer Education Techniques, Inc.

1: 21

PATH - Setting

- When the following command is typed:
\$ date
- The shell has to locate the command date before it can be executed.
- The PATH specifies the locations in which the shell should look for commands.
- Typically it will be set as:
PATH=/bin:/usr/bin
- Each of the individual entries separated by the colon character, :, are directories.

SYS-ED/Computer Education Techniques, Inc.

1: 22

Fundamentals

UNIX Shell Programming

Shell Script – Making Executable

- An important task in writing shell scripts is to make the shell script executable and ensure that the correct shell is invoked on the script.
- In a previous example, the logins script executed the following compound command:
`date ; who ;`
- In order to execute the script by typing its name, it will be necessary to:
 - Make it executable.
 - Ensure that the correct shell is used when the script is run.
- In order to make this script executable, perform the following:
`chmod a+x ./logins`
- In order to ensure that the correct shell is used to run the script, it will be necessary to add the following "magic" line to the beginning of the script:
`#!/bin/sh`
- The script then has two lines:
`#!/bin/sh`
`date ; who ;`

SYS-ED/Computer Education Techniques, Inc.

1: 23

Comments

- The magic first line `#!/bin/sh` is a comment.
 - A comment is a statement that is embedded in a shell script but should not be executed by the shell.
- In shell scripts, comments start with the `#` character.
 - Everything between the `#` and end of the line are considered part of the comment and are ignored by the shell.
- Adding comments to a script is straightforward:
 - Open the script using an editor and add lines that start with the `#` character.
 - For example, to add the following line to the logins shell script:
 - `# print out the date and who's logged on`
- The file `logins` was opened with an editor and the line was inserted as the second line in the file.
- The shell script is now:
`#!/bin/sh`
`# print out the date and who's logged on`
`date ; who ;`

SYS-ED/Computer Education Techniques, Inc.

1: 24

Fundamentals

UNIX Shell Programming

Getting Help

- Every version of UNIX comes with an extensive collection of online help pages called manual pages.
 - These are commonly referred to as man pages.
- The man pages are the authoritative source about the UNIX system.
 - They contain complete information about both the kernel and all the utilities.
- In order to access a man page, it will be necessary to use the man (man as in manual) command:
 - man command
- The command is the name of a command for which additional information is required.
- For example:
 - \$ man uptime

SYS-ED/Computer Education Techniques, Inc.

1: 25

Working with Files

Fundamentals

UNIX Shell Programming

File Types

- There are three basic types of files:
 - Ordinary
 - Directories
 - Special
- An ordinary file is a file on the system that contains data, text, or program instructions.
- Special files provide access to hardware such as hard drives, CD-ROM drives, modems, and Ethernet adapters.
 - Other special files are similar to aliases or shortcuts and enable access to a single file using different names.

SYS-ED/Computer Education Techniques, Inc.

1: 27

Listing Files

- The following command can be used for listing files and directories stored in the current directory.

```
$ ls
```

- This output indicates that several items are in the current directory; however, this output does not indicate whether these items are files or directories

- In order to ascertain which of the items are files and which are directories, the -F option has to be specified.

```
$ ls -F
```

SYS-ED/Computer Education Techniques, Inc.

1: 28

Fundamentals

UNIX Shell Programming

Listing Files

- When the -F option is specified to ls, it appends a character indicating the file type of each of the items it lists.
 - The exact character depends on the version of ls.
 - For ordinary files, no character is appended.
 - For special files, a character such as !, @, or # is appended to the filename.
- In a shell script it is much easier to manipulate the output when each file is listed on a separate line.
- The ls command supports the -1 option to do this.
 - For example:
\$ ls -1

SYS-ED/Computer Education Techniques, Inc.

1: 29

Hidden Files

- Up to this point, we have made extensive use of the ls to list visible files and directories.
- However, ls can also list invisible or hidden files and directories.
 - An invisible file has the first character as the dot or period character (.).
 - UNIX programs (including the shell) use most of these files to store configuration information.
- In order to list invisible files, specify the -a option to ls:
\$ ls -a
- In order to get the file type information, specify the -F and the -a options as follows:
\$ ls -a -F

SYS-ED/Computer Education Techniques, Inc.

1: 30

Fundamentals

UNIX Shell Programming

Option Grouping

- In the previous example, the command was used for specifying the options to ls separately.
- These options can also be grouped together.
 - For example
 - The commands:
\$ ls -aF
\$ ls -Fa
 - Provide the identical functionality as the command:
\$ ls -a -F

SYS-ED/Computer Education Techniques, Inc.

1: 31

Content of a File - Viewing

- Shell scripts need to be able to view the contents of a file.
cat
- In order to view the content of a file, use the cat (short for concatenate) command.
- Its syntax is:
cat files
 - The files are the names of the files that are to be viewed.
 - For example:
\$ cat hosts
 - Prints out the contents of a file called hosts
- More than one file can be specified.
\$ cat hosts users

SYS-ED/Computer Education Techniques, Inc.

1: 32

Fundamentals

UNIX Shell Programming

Numbering Lines

- The cat command also understands several options.
 - The -n option numbers the output lines.
 - It can be used as follows:
\$ cat -n hosts
- The cat command can be used for skipping numbering blank lines using the -b option:
 - \$ cat -b hosts

SYS-ED/Computer Education Techniques, Inc.

1: 33

wc Command - Counting Words

- The wc command can be used for getting a count of the total number of lines, words, and characters contained in a file.
- The command syntax is:
wc [options] files
- If no options are specified, the output contains a summary of the number of lines, words, and characters.
- If more than one file is specified, wc gives the individual counts along with a total.
 - For example, the command:
\$ wc .rhosts .profile
 - -l Counts the number of lines.
 - -w Counts the number of words.
 - -m or -c Counts the number of characters.

SYS-ED/Computer Education Techniques, Inc.

1: 34

Fundamentals

UNIX Shell Programming

Manipulating Files

cp Command - Copying Files

- In order to make a copy of a file use the `cp` command.
- The command syntax is:
`cp source destination`
- The source is the name of the file that is copied and destination is the name of the copy.
 - For example, the following command makes a copy of the file `test_results` and places the copy in a file named `test_results.orig`:
`$ cp test_results test_results.orig`

Fundamentals

UNIX Shell Programming

Common Errors

- There is no output from the cp command, unless it encounters an error.
- Two common errors occur when:
 - The source is a directory.
 - The source does not exist.

SYS-ED/Computer Education Techniques, Inc.

1: 37

Interactive Mode

- No error message is generated if the destination already exists.
 - In this case, the destination file is automatically overwritten.
 - This can lead to serious problems.
- In order to avoid this behavior, the -i options for the cp command can be used.
 - If the file test_results.orig exists, the command:
\$ cp -i test_results test_results.orig
 - results in a prompt along the lines such as:
overwrite test_results.orig? (y/n)
- If y (yes) is chosen, the file will be overwritten.
- If n (no) is chosen, the file test_results.orig isn't changed.

SYS-ED/Computer Education Techniques, Inc.

1: 38

Fundamentals

UNIX Shell Programming

Copying Files to a Different Directory

- If the destination is a directory, the copy has the same name as the source however it is located in the destination directory.
- For example, the command:
\$ cp test_results work/
 - Places a copy of the file test_results in the directory work.

SYS-ED/Computer Education Techniques, Inc.

1: 39

Multiple Inputs

- If more than two inputs are given, cp treats the last argument as the destination and the other files as sources.
 - This works only if the sources are files and the destination is a directory, as in the following example:
\$ cp res.01 res.02 res.03 work/
- If one or more of the sources are directories the following error message is produced.
 - For example:
\$ cp res.01 work/ docs/ pub/
produces the following error:
cp: work: is a directory
cp: docs: is a directory
- Although cp reports errors, the source file, in this case res.01, is correctly copied to the directory pub.

SYS-ED/Computer Education Techniques, Inc.

1: 40

Fundamentals

UNIX Shell Programming

mv Command - Renaming Files

- The mv command is used for changing the name of a file.
 - The syntax is:
mv source destination
 - The source is the original name of the file.
 - The destination is the new name of the file.
 - For example:
\$ mv test_result test_result.orig
- The name of the file test_result is changed to test_result.orig.
- A new file called test_result.orig is not produced as with the cp command; only the name of the file is changed.

SYS-ED/Computer Education Techniques, Inc.

1: 41

mv Command - Renaming Files

- No output from the mv command indicates that the name change is successful.
- If the source does not exist, as in the following example,
\$ mv test_reslut test_result.orig
- An error similar to the following is reported:
mv: test_reslut: cannot access: No such file or directory

SYS-ED/Computer Education Techniques, Inc.

1: 42

Fundamentals

UNIX Shell Programming

Interactive Mode

- As with `cp`, `mv` does not report an error if the destination already exists; it will overwrite the file.
- The `-i` option can be specified in order to avoid this problem.
 - For example, if the file `ch07.bak` already exists, the following command:
 - `$ mv -i ch07 ch07.bak`
 - results in a confirmation prompt:
remove `ch07.bak`? (n/y)
- If `n` (no) is chosen, the destination file is not touched.
- If `y` (yes) is chosen, the destination file is removed and the source file is renamed.
- The actual prompt varies between the different versions of UNIX.

SYS-ED/Computer Education Techniques, Inc.

1: 43

rm Command - Removing Files

- In order to remove files, use the `rm` command.
- The syntax is:
 - `rm files`
- `Files` is a list of one or more files to remove.
 - For example:
 - The command:
 - `$ rm res.01 res.02`
 - removes the files `res.01` and `res.02`.

SYS-ED/Computer Education Techniques, Inc.

1: 44

Fundamentals

UNIX Shell Programming

Interactive Mode

- Since there is no way to recover a file that has been deleted using `rm`, the `-i` option can be specified.
- In interactive mode, `rm` prompts for every file that is requested for deletion.
 - For example:
 - The command:
`$ rm -i hw1 hw2 hw3`
 - produces confirmation prompts similar to the following:
hw1: ? (n/y) y
hw2: ? (n/y) n
hw3: ? (n/y) y
- In this case, the answer is to delete `hw1` and `hw3`, but not to delete `hw2`.

SYS-ED/Computer Education Techniques, Inc.

1: 45

Directories

Fundamentals

UNIX Shell Programming

Working with Directories

- UNIX uses a hierarchical structure for organizing files and directories.
 - This structure is referred to as a directory tree .
 - The tree has a single root node, the slash character (/), and all other directories are contained below it.
- Every directory, including /, can be used for storing both files and other directories.
- Every file is stored in a directory, and every directory except / is stored in another directory.

SYS-ED/Computer Education Techniques, Inc.

1: 47

Filenames

- In UNIX, every file and directory has a name associated with it. This name is referred to as the file or directory's filename.
- In addition to their filenames, every file and directory is associated with the name of its parent directory. When a filename is combined with the parent directory's name, the result is called a pathname. Two examples of pathnames are
 - /home/ranga/docs/book/ch5.doc
 - /usr/local/bin/
- As you can see, each of these pathnames consists of several "words" separated by the slash (/) character. In UNIX, the slash separates directories, whereas the individual words are the names of files or directories. The sum of all the words and the / characters makes up the pathname.

SYS-ED/Computer Education Techniques, Inc.

1: 48

Fundamentals

UNIX Shell Programming

Filenames

- The last set of characters in a pathname is the actual name of the file or directory being referenced.
- The rest of the characters represent its parent directories.
- In the first example, the filename is ch5.doc.
- The name of a file can be up to 255 characters long and can contain any ASCII character except /.
- Generally, the characters used in pathnames are the alphanumeric characters (a to z, A to Z, and 0 to 9) along with periods (.), hyphens (-), and underscores (_).

SYS-ED/Computer Education Techniques, Inc.

1: 49

Pathnames

- In order to access a file or directory, its pathname must be specified.
- A pathname consists of two parts:
 - the name of the directory.
 - the names of its parents.

SYS-ED/Computer Education Techniques, Inc.

1: 50

Fundamentals

UNIX Shell Programming

Pathnames

- UNIX offers two ways to specify the names of the parent directory.
- This leads to two types of pathnames:
 - Absolute
 - Relative

SYS-ED/Computer Education Techniques, Inc.

1: 51

Absolute Pathnames

- An absolute pathname represents the location of a file or directory starting from the root directory and listing all the directories between the root and the file or directory of interest.
- Since absolute pathnames list the path from the root directory, they always start with the slash (/) character.
 - Regardless as to what the current directory is, an absolute path points to an exact location of a file or directory.
 - Example: Absolute pathname

`/home/ranga/work/bugs.txt`
- This absolute path provides the information that:
 - the file bugs.txt is located in the directory work.
 - which is located in the directory ranga.
 - which in turn is located in the directory home.
- The slash at the beginning of the path provides the information that the directory home is located in the root directory.

SYS-ED/Computer Education Techniques, Inc.

1: 52

Fundamentals

UNIX Shell Programming

Relative Pathnames

- A relative pathname enables to access files and directories by specifying a path to that file or directory within that current directory.
- When the current directory changes, the relative pathname to a file can also change.
- The pwd (print working directory) command can be used for ascertaining the current directory. It prints the name of the directory which is the current working directory.
 - For example:
\$ pwd
/home/ranga/pub
 - Provides the information which indicates that the user is located in the the directory /home/ranga/pub.

SYS-ED/Computer Education Techniques, Inc.

1: 53

Relative Pathnames

- When specifying a relative pathname, the slash character is not present at the beginning of the pathname.
- This indicates that a relative pathname is being used instead of an absolute pathname.
- The relative pathname is a list of the directories located between the current directory and the file or directory that is being represented

SYS-ED/Computer Education Techniques, Inc.

1: 54

Fundamentals

UNIX Shell Programming

Home Directories

- First print the working directory:
\$ pwd
/home/ranga
- This indicates that the user is in his/her home directory.
 - The home directory is the initial directory where starting when logging in to a UNIX machine.
 - Most systems use either /home or /users as directories under which home directories are stored. On my system I use /home.

SYS-ED/Computer Education Techniques, Inc.

1: 55

Home Directories

- The easiest way to determine the location of a home directory is to run the following sequence of commands:
\$ cd
\$ pwd
/home/ranga
- When issuing the cd command without arguments, it changes the current directory to the home directory.
- Therefore, after the cd command completes; the pwd command prints the working directory that is the user's home directory.

SYS-ED/Computer Education Techniques, Inc.

1: 56

Changing Directories

- The `cd` command can be used to do more than changing to a home directory.
- It can be used for changing to any directory by specifying a valid absolute or relative path.
- The syntax is as follows:
`cd directory`
 - The `directory` is the name of the directory that is to be changed to.

SYS-ED/Computer Education Techniques, Inc.

1: 57

Changing Directories

For example:

- The command:
`$ cd /usr/local/bin`

changes to the directory `/usr/local/bin`. Here, you used an absolute path.

- Say that the current directory is:
`$ pwd`
`/home/ranga`

SYS-ED/Computer Education Techniques, Inc.

1: 58

Fundamentals

UNIX Shell Programming

Listing Files and Directories

- In order to list the files in a directory, the following syntax can be used:
 ls directory
- The directory is the absolute or relative pathname of the directory whose contents are to be listed.
- More than one directory can be specified as an argument.
- For example:
 \$ ls /home /usr/local

SYS-ED/Computer Education Techniques, Inc.

1: 59

Manipulating Directories

- The most common manipulations are:
 - Creating directories
 - Copying directories
 - Moving directories
 - Removing directories

SYS-ED/Computer Education Techniques, Inc.

1: 60

Fundamentals

UNIX Shell Programming

Listing Files

- If the name of the file is specified instead of a directory, ls lists only that one file.
 - For example:
\$ ls .profile
.profile
- Files and directories can be submitted together as arguments to a single ls command:
 - For example:
\$ ls .profile docs/ /usr/local /bin/sh
 - This command produces a listing of the specified files and the contents of the directories.
- The `-d` option to ls can be used for suppressing the listing of the directory to be displayed. Only the name of the directory will be displayed; not its contents.

SYS-ED/Computer Education Techniques, Inc.

1: 61

Creating Directories

- Directories can be created with the mkdir command.
 - The command syntax is:
 - mkdir directory
- The directory is the absolute or relative pathname of the directory that is to be created.
- Example 1:
 - The command:
\$ mkdir hw1
 - Creates the directory hw1 in the current directory.

SYS-ED/Computer Education Techniques, Inc.

1: 62

Fundamentals

UNIX Shell Programming

Creating Directories

- Example 2:
\$ mkdir /tmp/test-dir
 - This command creates the directory test-dir in the /tmp directory.
 - The mkdir command produces no output if it successfully creates the requested directory.
- If more than one directory is presented on the command line, mkdir creates each of the directories.
- Example 3:
\$ mkdir docs pub
 - creates the directories docs and pub under the current directory.

SYS-ED/Computer Education Techniques, Inc.

1: 63

Creating Parent Directories

- Frequently it will be necessary to create a directory, its parent directory or directories might not exist.
 - In this case, mkdir issues an error message.
 - For example:
\$ mkdir /tmp/ch04/test1
 - mkdir: Failed to make directory "/tmp/ch04/test1"; No such file or directory
- In such cases, the -p command option can be specified (p as in parent) option to the mkdir command.
- It will create all the necessary directories.
 - For example:
\$ mkdir -p /tmp/ch04/test1
 - creates all the required parent directories.

SYS-ED/Computer Education Techniques, Inc.

1: 64

Fundamentals

UNIX Shell Programming

Copying Files and Directories

- In order to copy a directory, the -r option to cp needs to be specified.
- The syntax is:
 - cp -r source destination
 - The source is the pathname of the directory to be copied.
 - The destination is where the copy is to be placed.

SYS-ED/Computer Education Techniques, Inc.

1: 65

Copying Files and Directories

- For example:
 - \$ cp -r docs/book /mnt/zip
 - Copies the directory book located in the docs directory to the directory /mnt/zip.
 - It creates a new directory called book under /mnt/zip.

SYS-ED/Computer Education Techniques, Inc.

1: 66

Fundamentals

UNIX Shell Programming

Copying Multiple Directories

- It is also possible to copy multiple directories.
- If `cp` encounters more than one source, all the source directories are copied to the destination. The destination is assumed to be the last argument.
- For example:
 - The command:
`$ cp -r docs/book docs/school work/src /mnt/zip`
 - Copies the directories `school` and `book`, located in the directory `docs`, to `/mnt/zip`. It also copies the directory `src`, located in the directory `work`, to `/mnt/zip`.

SYS-ED/Computer Education Techniques, Inc.

1: 67

Moving Files and Directories

- Although the `mv` command can be used for renaming files, it is also has the capability to move files and directories between different locations in the directory tree.
- The command syntax is:
`mv source destination`
- The source is the name of the file or directory to be moved.
- The destination is the directory where the file or directory is to end up.
- For example:
 - `$ mv /home/ranga/names /tmp`
 - moves the file `names` located in the directory `/home/ranga` to the directory `/tmp`.

SYS-ED/Computer Education Techniques, Inc.

1: 68

Fundamentals

UNIX Shell Programming

Moving Files and Directories

- Moving a directory is exactly the same:
\$ mv docs/ work/
- moves the directory docs into the directory work. To move the directory docs back to the current directory you can use the command:
\$ mv work/docs .
- One nice feature of mv is that you can move and rename a file or directory all in one command.

SYS-ED/Computer Education Techniques, Inc.

1: 69

Removing Directories

- There are two commands for removing directories:
rmdir
rm -r
- The first command is used for removing empty directories.
- It is considered "safe" since in a worst case scenario, when an empty directory is lost, it can be quickly re-created with mkdir.
- The rmdir command removes directories along with its contents. It is considered "unsafe" because in a worst case use rm -r, an entire system could be lost.

SYS-ED/Computer Education Techniques, Inc.

1: 70

Fundamentals

UNIX Shell Programming

Removing Directories

- In order to remove an empty directory, the `rmdir` command can be used.
- The syntax is:
`rmdir directories`
 - The directories include the names of the directories to be removed.
 - For example, the command:
`$ rmdir ch01 ch02 ch03`
 - Removes the directories `ch01`, `ch02`, and `ch03` if they are empty.
 - The `rmdir` command produces no output when it is successful.