CST 126 – LESSON 3

Touring Essential Programs

Overview

- Employing fundamental utilities.
- Managing input and output.
- Using special characters in the command-line.
- Managing user environment.
- Surveying elements of a functioning system.
- Creating a shell script.

Employing Fundamental Utilities

- Listing the contents of a directory.
- Counting the elements of a file.
- Sorting lines in a file.
- Passing arguments to utilities.
- Creating combination files.
- Locating specific lines in a file.

Listing the Contents of a Directory

- Each utility is a tool that performs a set of very specific tasks.
- The "ls" utility outputs the filenames listed in the current directory.
- The "–F" option with ls places a forward slash at the end of the name of each directory.
- The "–F" option places an asterisk (*) at the end of the executable files.

Listing the Contents of the Directory

- The "ls –l" command displays a file's permissions and other data about the file.
- The "–a" (all) option to ls includes all files in the current directory, including dot or hidden files.
- The "ls –alF" command displays all files with the long listing of information, and directories marked with a slash.

Counting the Elements of a File

- The "wc" command counts the number of lines, words, and characters in a file.
- It operates on the files individually.
- It outputs the statistics for each file and then produces a total.

OPTION	OUTPUT
wc -l *users_on*	The count of lines only.
wc -w *users_on*	The count of words only.
wc -c *users_on*	The count of characters only.

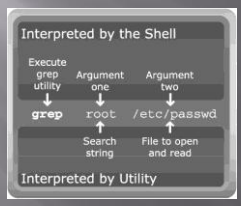
Sorting Lines in a File

- The /etc/passwd (the password file) contains one line of information (a record) for each user.
- The "sort" utility reads the file into memory and rearranges the lines into a sorted order.
- In the sort order, lines beginning with numbers are displayed first, lines beginning with uppercase letters second, and lines beginning with lowercase letters last.
- The "–r" (reverse) option is used to sort a file in reverse ASCII order.

Passing Arguments to Utilities

- When an argument is passed to a utility, that utility's code determines how that argument is interpreted.
- Passing arguments to utilities provides an opportunity to let the shell read the base command.
- Passing arguments also provides the output as requested by the arguments provided to the utility.

Locating Specific Lines in a File



Using the grep Utility to Locate Specific Files

Managing Input and Output

- Every process that is started has three defined communication locations or doors.
- One is its input, the second to write output, and the third to write any error messages. (old terminology: stdin, stdout, stderr)

COMMAND	INTERPRETATION
utility > *filename*	Shell connects the output of the utility to *filename*.
utility >> *filename*	Shell connects the output of the utility to the end of the file (appends).
utility < *filename*	Shell connects *filename* to the input of the utility.
utility1 | utility2	Shell connects the output of **utility1** to the input of **utility2**.

Redirection Operators

Managing Input and Output with Redirection

COMMAND	INPUT	OUTPUT	EFFECT
sort	Keyboard	Display screen	The **sort** utility receives no arguments, so it opens no file. Instead, **sort** reads from input, which is by default connected to the keyboard.
sort > *file1*	Keyboard	*file1*	Keyboard input is sorted, and output is connected to *file1*.
sort >> *file1*	Keyboard	*file1*	Keyboard input is sorted and its output is appended to the end of *file1*.
sort < *file2*	*file2*	Display screen	*file2* is opened by the shell and connected to the input of **sort**. The output is not redirected, but displayed on the screen.

Passing Arguments and Opening Files

Managing Input and Output with Redirection

COMMAND	INPUT	OUTPUT	EFFECT
sort *file1*	*file1*	Display screen	*file1* is passed as an argument to **sort**, which opens the file and reads its contents. Output connected to the screen.
sort < *file1* **>** *file3*	*file1*	*file3*	The shell connects *file1* to the input and *file3* to the output of **sort**. When **sort** runs, it reads from input, sorts the lines, and writes to output. The lines from the file *file1* are sorted and the output placed in *file3*.

Passing Arguments and Opening Files

Creating Text Files with cat

- The "cat" command is used to create small text files without using an editor.
- The cat utility reads the input from the user and writes it to its output.
- The Ctrl-D (end-of-file, or EOF) key combination is used to tell cat that there is no additional input.
- By default, the keyboard is connected to the input of cat.

Using Special Characters in the Command-Line

- The shell interprets the > (redirect) instruction to connect the output of the previous utility to a file named right after the redirection.
- The | (pipe) is the instruction to connect the output of one utility to the input of another.
- A wildcard character is a special character to the shell when specifying filenames.
- The shell replaces the asterisk (*) with the names of all the files in the directory and then executes the command as requested.

Using Special Characters in the Command-Line

- Accessing shell variables.
- Listing environment variables.
- Interpreting special characters.
- Creating multiple token arguments.
- Passing complex arguments.
- Communicating with processes.
- Programming with utilities.

Accessing Shell Variables

- The "$" character carries a special meaning for the shell.
- The $ variable instructs the shell to locate the variable whose name follows, and replace the string with the variable's value.

Listing Environment Variables

- When a user logs on, the shell program that interprets the commands is started.
- The values of several variables are given to the shell so that the computing environment is appropriate.
- The "env" and the "printenv" command displays a list of the variables that are currently set for the shell.
- Variables and their values are essential to a functioning shell.

Listing Environment Variables

Some of the more well-known environment variables available in UNIX/Linux are:

- The user or USER or LOGNAME variable is the account name entered by the user while logging on to the system.
- The shell or SHELL variable indicates which of the shell programs is started at login.
- The home or HOME variable is the location of the user's home directory.
- The path or PATH variable lists the directories used by shell to find UNIX utilities.

Interpreting Special Characters

- The characters *, !, |, >, and $ have special meaning to the shell.
- The shell is sometimes instructed not to interpret special characters, but to treat them as ordinary characters instead.
- The shell interprets "enter" as a special character, one that indicates the end of a command to be executed and start processing.

Interpreting Special Characters

- The backslash before the Enter command stops the shell from reading the enter keystroke. (ex. \<enter>)
- The backslash turns off interpretation for one character only.
- The interpretation for multiple characters can be turned off using single quotes. (ex. echo '$HOME')
- Any special characters inside single quotes are not interpreted.

Multiple Token and Complex Arguments

- The shell interprets one or more spaces as separating the tokens on the command-line.
- Single quotes are used to inform the shell not to interpret the spaces.
- The UNIX operating system provides several utilities that are used with database information.
- The "awk" utility is used to select and print specific fields, make calculations, and locate records by the value of specific fields.

Communicating with Processes

- The control character Ctrl-D is used to signal the end of file or input.
- The control character Ctrl-C is the interrupt signal that kills a process.
- The ampersand (&) at the end of a command tells the shell to execute the whole command line in the background.
- The & instructs the shell to return a new shell prompt for the user to continue working, instead of waiting for the sleep command to finish execution.

Programming with Utilities

- The shell can read instructions that are placed in a file.
- The "source" command in the csh or tcsh shell can be used to instruct the current shell to read a file and execute each line in a file.
- The . (dot) command in the ksh, bash, or sh can be used to instruct the current shell to read a file and execute each line in a file.

Managing User Environment

- To avoid overwriting overwriting files, the user needs to set the "noclobber" option.

SHELL	TURN *noclobber* ON	TURN *noclobber* OFF
csh, tcsh	set *noclobber*	unset *noclobber*
ksh, bash	set -o *noclobber*	set +o *noclobber*

- To avoid accidental logouts, the user needs to instruct the shell to ignore an end-of-file character.
- In the csh or tcsh shells, "set ignoreeof".
- In the ksh and bash shells, "set -o ignoreeof".

Managing User Environment

- In a csh or tcsh shell, the command "set prompt='myprompt '" is used to set the prompt.
- In the sh, bash, or ksh shells, the command "PS1='myprompt '" is used to set the prompt.

PROMPT	SHELL
$	Bourne and Korn shells (**sh**, **bash**, and **ksh**)
%	C shells (**csh** and **tcsh**)
#	Any shell as ***root***

Surveying Elements of a Functioning System

- The variable PATH consists of a series of directories separated by a colon.
- The /bin directory contains some of the utilities available on the system in the form of binary files.
- The "which" command is used to determine the location of a utility.

Surveying Elements of a Functioning System

- When a user logs in to a system, the entry from either the local or the NIS network passwd file is consulted.
- The records in the /etc/passwd file consist of seven fields separated by colons.

donaldsouthwell:x:10996:700:Donald Southwell:/usr/users/donaldsouthwell:/bin/ksh

- The "ypcat" command is used to see the passwd file information from the network server that relies on NIS.

Surveying Elements of a Functioning System

donaldsouthwell:x:10996:700:Donald Southwell:/usr/users/donaldsouthwell:/bin/ksh

FIELD	INFORMATION
login	The login or name for your account.
password	Your encrypted password. (May be an *x* if the passwords are kept in a secure ***/etc/shadow*** file. May also be an *.)
uid	Your user ID, the unique number that is assigned to your account.
gid	Your group ID. Each user must be a member of at least one group. Every user who has the same number in this field as you have is in your group. You can share files with group members using permissions.
misc	Information about the user such as the user's full name. The miscellaneous field is often blank.
home	Your home directory. This is your current directory when you first log on.
Startup program	The program that is started when you log on—it is usually a shell such as the Bash shell (***/usr/bin/bash***) or the **tcsh** (***/bin/tcsh***) or the Korn shell (***/bin/ksh***), but it does not have to be a shell. It can be anything, including a data entry program or a menu for accessing your accounts at a bank.

The Fields of the password File

Surveying Elements of a Functioning System

- The "yppasswd" command is used to change a user's password on a NIS system.
- The "passwd" command is used to change the user's password on a normal system.
- The root user can change the user's password to a new one without knowing the original.

Surveying Elements of a Functioning System

- The owner of a file determines who has permissions to read or change its contents.
- All files have a set of permissions that determine who can do what with the file.
- The minus (-) sign is used to remove any permission on the file using the chmod command.
- The plus (+) sign is used to add any permission on the file using the chmod command.

Creating a Shell Script

The steps to create and use a shell script are:

- Create a file of shell commands.
- Make the file executable with chmod.
- Execute the file by entering the script name.

• Read permission is required to source a script.
• Execute permission is needed to start a child shell to execute its contents.

Summary

- Utilities are essential tools for accomplishing work in UNIX/Linux.
- The shell redirects input and output for processes running utilities. Redirection is used extensively in command line processing.
- The shell interprets special characters differently to support enhanced functionality.
- The shell supports user scripting and options to control access and execution.