

Fundamentals of UNIX
Lab 14.2.5 – Korn Shell Features
(Estimated time: 45 min.)

Objectives:

- Develop an understanding of Korn shell features
- Review the shell
- Review aliases
- Create aliases
- Display aliases
- Remove and bypass an alias
- Display command history
- Re-execute commands
- Edit the command line
- Customize your shell prompt

Background:

In this lab, you will work with the **Korn** shell to understand its features and capabilities. The **shell** is the primary user interface to a UNIX system. The concept of the shell was introduced in Chapter 1 along with other key UNIX operating system components: the **kernel** and the **file system**. The UNIX environment provides support for many built-in (i.e. **Bourne, C, Korn**) and third party shells. This lab provides a brief review of the function of the shell and focuses on the most popular shell used with UNIX systems today: the Korn Shell. It goes into greater detail about the unique features of the Korn shell and provides opportunities to become a more efficient user. You will work with aliases, history, and re-execution of commands and custom prompts in this lab.

Tools / Preparation:

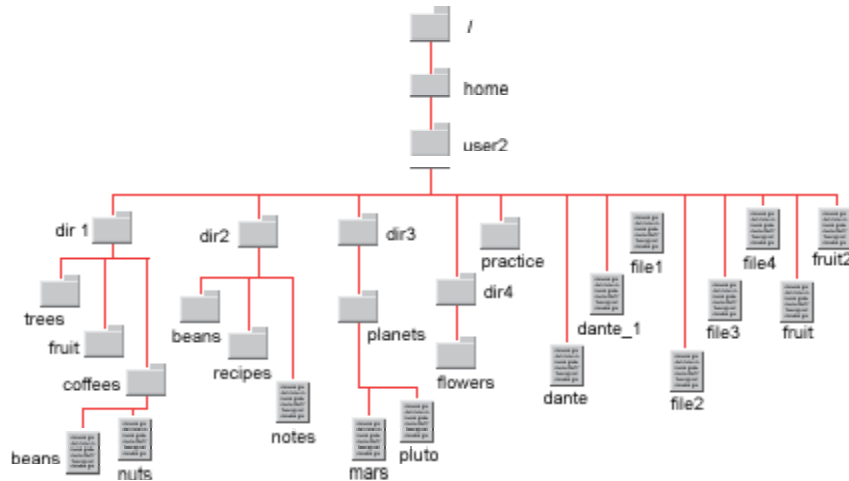
- a) Before starting this lab, review Chapter 14, Section 1 – Review of the Shell and Section 2 – Korn Shell Features
- b) You will need the following:
 1. A login user ID (e.g. user2) and password assigned by your instructor.
 2. A computer running the UNIX operating system
 3. Networked computers in classroom with class file system installed

Notes:

Worksheet

Use the diagram of the sample Class File System directory tree to assist with this lab.

Class File Tree Structure



Step 1. Log in to CDE

Login with the user name and password assigned to you by your instructor in the CDE entry box.

Step 2. Access the Command Line

Right click on the **workspace** backdrop and click on **Tools**. Select **Terminal** from the menu to open a terminal window.

Step 3. Review the Shell

A shell is an interface between the user and the kernel. It acts as an interpreter or translator. In other words, the shell accepts commands issued by you, interprets these commands, and executes the appropriate programs. Shells can be command-line driven or Graphical. The system administrator decides which shell will be the default for a user when they define the user ID. This lab will focus on the **Korn** (ksh) the most widely used shell for end users. You can use the **ps** (**process status**) command to see which shell you are using.

Bourne shell (\$) - The Bourne shell was the original shell program for the UNIX environment. It is the default shell for the Solaris computing environment. Stephen Bourne developed the Bourne shell for the AT&T System V.2 UNIX environment. This shell does not support the alias or history commands or command line editing capabilities and is used primarily by system administrators. The Bourne shell prompt is a dollar sign (\$).

Korn shell (\$) - The Korn shell is a superset of the Bourne shell and was developed by Stephen Korn at Bell Labs. It has many of the Bourne shell features plus added features such as aliasing, history, and command line editing. This is the most widely used shell and is the industry standard for system users. The Korn shell prompt is also a dollar sign (\$).

Note - Examples given in this course are based primarily on the Korn shell.

C shell (%) - A shell based on the C programming language. Like the Korn shell, it has additional features such as aliasing and history. The C shell was developed by Bill Joy of Sun Microsystems and is still widely used today. The C shell prompt is a percent sign (%).

Fundamentals of UNIX
Lab 14.2.5 – Korn Shell Features
Worksheet – Cont.

Using the information above fill in the blanks in the following sentences.

- a. A shell acts as an **interpreter** or _____ between the user and the kernel.
- b. The default shell for a typical user is decided by the _____
- c. You can determine which type of shell you are using by issuing the _____ command
- d. The _____ is based on the C programming language and was developed by Bill Joy of Sun Microsystems. This shell uses the percent sign (%) as a prompt
- e. The _____ shell was the original shell and does not support aliasing or history?
- f. The _____ shell uses the dollar sign (\$) as a prompt and supports aliasing, history and command line editing?

Step 4. Review Aliases

An **alias** is a way to give a command a different name for use in the shell. Aliases provide an excellent way to improve efficiency and productivity when using shell commands. When set from the command line, aliases are only activated for the shell in which they are created. Adding aliases to your **.kshrc** file (covered in a later lab) will activate them upon login or whenever a new window or shell is opened.

Command Format: **alias** **aliasname=value**

The reasons to use aliases are summarized below, along with some examples of how to create them. The syntax shown here is for the Korn shell only.

Substitute a short command for a long one - You can reduce the number of keystrokes for commonly used long commands by creating an alias for the command.

Example: \$ **alias** **c=clear**

Create a single command for a series of commands - You can string several commands together and assign them one short alias name to reduce keystrokes.

Example: \$ **alias** **home='cd;ls'**

Create alternate forms of existing commands - Some commands such as **rm** (remove files and directories) and **cp** (copy files) can be dangerous. An alias will allow you to change the meaning of these commands to include the **-i** option so the user is prompted before accidentally overwriting a file or directory.

Example: \$ **alias** **copy='cp -i'**

Use single quotes for commands with options, spaces or other special characters. There are no spaces between the **alias** command, the equal sign (=), and the command(s) being assigned to the alias.

Using the information above, fill in the blanks in the following sentences.

- a. An alias is a way to give a _____ a different name for use with the shell and can result in improved productivity.
- b. When set from the command line, aliases are only activated for the _____ in which they are created.

Fundamentals of UNIX
Lab 14.2.5 – Korn Shell Features
Worksheet – Cont.

- c. Use _____ for commands with options, spaces or other special characters.
- d. There are no _____ between the alias command, the equal sign (=), and the command(s) being assigned to the alias.

Step 5. Create Aliases

- a. Create an alias called **h** (lower case letter h) to substitute for the Korn shell **history** command. What command did you use? _____
- b. Try the new **h** alias. What does it do? _____
- c. Create an alias for the **ps -ef** command called **p** (lower case letter p) that will **pipe** the **more** command to the end of the **ps -ef** command. What command did you use? _____
- d. Try the new **p** alias. What does it do? _____
- e. Create an mv alias that will substitute the command **mv -i** command for the **mv** command to prevent accidentally overwriting files when moving them. What command did you use?

- f. Try the new **move** alias. Copy the dante file into your practice directory first and then attempt to move dante to the practice directory using the **move** alias. What does it do?

Step 6. Display Aliases

To display aliases, use the **alias** command with no arguments/options. Using the **alias** command by itself will display all aliases set for the current session. Some aliases are pre-defined with the Korn shell.

- a. Display all aliases for your current session. Do you see the ones you created previously?

- b. Use the **alias** command and pipe the output to the **wc -l** command. This will count the **lines** of output from the **alias** command. Subtract the ones you created from the total. How many other aliases were predefined as part of your Korn shell? _____

Step 7. Remove and Bypass an Alias

You can **unset** a previously defined alias with the **unalias** command. The alias will no longer appear in the alias listing. You can also temporarily bypass an Alias. To bypass the alias and use the original version of a command, use a backslash before the command.

Command Format: ***unalias aliasname***

- a. Remove or unset the **h** alias you defined previously. What command did you use? _____
- b. Display all aliases for your current session. Do you see the h alias? _____
- c. Use the **bypass** option to bypass the **mv** alias you created previously and use the original version of the **mv** command to move the dante file from the practice directory to your home directory. Were you prompted before overwriting? _____

Fundamentals of UNIX
Lab 14.2.5 – Korn Shell Features
Worksheet – Cont.

- d. Type **exit** at the command line to close your current terminal window and then start another new terminal window. Enter the **alias** command to list currently defined aliases for this session. Are any of the aliases you defined previously there now? _____ Why or why not?
-

Step 8. Display Command history

The **history** feature records commands typed in the Korn shell. In the Korn shell, **history** is automatically set up when the user first enters the shell. Whenever the user types in a command, this function records it on a history list. The Korn shell keeps a record of the last 128 commands entered but only displays the last 16 commands by default.

Command Format: ***history*** ***[options]***

Specifying a **number preceded by a dash** as an option will display that number of lines. Specifying a **number *without* a dash** will show the **history** lines from that number to the last command entered.

- a. Enter the **history** command by itself. What was the result? _____
- b. Display the last 5 lines of history by using the **-5** option. What was the last line displayed?
-

Step 9. Re-execute Commands

The **r** (repeat) command is one of many pre-defined aliases in the Korn shell. This enables the user to re-execute commands from the history list.

Command Format: ***r*** ***[argument(s)]***

The argument passed to the **r** command is the history line number of a particular command, or a letter indicating the most recent command in the history list that started with that letter. The **r** command by itself will repeat the last command entered (**pwd**). This is similar to the F3 key in DOS.

- a. As mentioned previously, the **r** command is a predefined Korn shell alias. Enter the **alias** command. Examine the alias listing, what command does the **r** alias actually execute? _____ If you wanted to know more about the **fc** command what could you do? _____
- b. Enter the **history** command. Identify the command number of one of your previous commands. What was the command number and command? _____
- c. Repeat that command using the repeat (**r**) command. Did the command you used earlier re-execute? _____
- d. Repeat a previous command that started with the letter (**a**). This will repeat alias command in this case. What command did you use? _____

Fundamentals of UNIX
Lab 14.2.5 – Korn Shell Features
Worksheet – Cont.

Step 10. Edit the Command Line

The in-line edit mode enables you to edit a previous command on the current command line. Use **vi** commands to move and edit the previous command line.

Command Format: **set [-+]*o* vi**

Using **set -o vi** turns command-line editing on, while **set +o vi** turns it off. Once the editing has been turned on, pressing the **Esc** key activates the in-line editor. You then have access to vi commands. The following table shows some of the most commonly used vi line editing commands. **Note** - The arrow keys cannot be used to reposition the cursor during in-line editing.

Command Line Edit Commands

vi Command	Meaning
k	Move backward through history list
j	Move forward through history list
l	Move to the right one character
h	Move to the left one character
i	Insert characters
r	Replace one character
cw	Change word
x	Delete one character

- a. Enter a command such as: **more dir1/coffees/beand** where the name of the file **beans** is misspelled. Press Enter to receive an error.
- b. Set the vi editor on so you can edit the command line. Press **Esc** to enter vi command mode. Press **k** to “kick up” through the command history until you get to the command with the error in it.
- c. Use the vi commands from the table above to move to the misspelled character in the command and press **r** to replace the d with the letter s. Press Enter to execute the command. Did it execute correctly this time? _____

Step 11. Customize your Shell Prompt

The Korn shell uses the Bourne shell as a basis for its features. These shared features include a login initialization file and variables (a place holder for information used for customizing the shell). In addition, the Korn shell provides aliasing, command tracking, command-line editing which you have used so far in this lab. These features allow the user to be more productive. The initialization file used to store this information is the **.kshrc** (Korn shell run control) file and is stored in the user's home directory.

The default prompt for the Korn shell is the dollar sign (\$). The user can customize their own shell prompt to their liking using the **PS1** (Prompt String 1) predefined shell prompt variable (PS1=\$). **PS1** can include a wide range of expressions such as character strings, commands, or other variables. The variable PS1 is a shell variable. Any change in the variable setting will remain until the shell is exited or until a subshell is opened. To make the customized prompt available from one session to the next, place the PS1 variable in the **.kshrc** file (covered in Chapter 15).

Fundamentals of UNIX
Lab 14.2.5 – Korn Shell Features
Worksheet – Cont.

Command Format: PS1=value

a. The following are several different ways the standard **PS1** prompt (\$) can be customized. Try each of these and then experiment with some of your own.

```
$ PS1="Good morning$ "
```

Assigns the prompt to a character string. Use one of your own.

```
$ PS1="\`uname -n` !$ "
```

Uses a command (**uname -n**) with the history line number (!) for a unique prompt. This command causes the prompt to show the name of the host on which you are working (Ex: **saturn41**). The back quotes (`) are used to substitute the output of the **uname** command instead of interpreting it literally. Double quotes surround the entire string.

```
$ PS1="$ "
```

Sets the prompt back to the original Korn shell dollar sign (\$) prompt with a space after it.

```
$ PS1="$PWD $"
```

The prompt will contain the current working directory. Double quotes surround the string and allow expansion of the **\$PWD** variable which keeps track of the current working directory.

Step 12 – Remove Files and Directories Created in this Lab

Remove all files and **directories** created in your home directory during this lab.

Step 13. Close the Terminal Window and Logout

Double click on the dash button in the upper left corner of the screen, then click the **EXIT** icon on the front panel.