Lab Assignment 9

**Title: Shell Programming-COPY UTILITY**

**Problem statement:**

To write a program to implement Shell Scripting (C program to copy) – Copy Utility

**Objectives:**

Understanding of Linux shell- Copy Commands.

**Theory:**

* **Linux Shells**

Linux comes with various command interpreters called as shells in the Unix terminology. Actually, shell sits in between the kernel of an operation system and the user. So whatever user wants to do through kernel it is available in terms of shell commands. Once you provide valid command for the required operation it hands over the request to the operating system kernel and finally job will be done by the system.

There are various shells available to use in the Linux environment but following shells are the standard shells.

Linux Shells

**Bourne Shell**

**C Shell**

**Korn Shell**

**Bash**

**tcsh**

The shells used in the Linux operating system has dual capability, in one had it is used as a tool which accepts commands interpret it and hands over it to the operating system kernel. Due to this capability it is called as command – line interpreter, another use of shell is it can be used as a programming language. Shell programming is interpretive by nature and mostly it is used to assist in system administration tasks.

* **Steps to interpret a shell script**

Assume a written shell script is stored in a script file named as Example-1. To execute this script we have two approaches: ***Make it as an executable and execute it as an argument to bash command.***

**Make it as an executable :**

* By the use of **chmod** command one can set execute permission on it. It is shown below for our first script.

**chmod +x Example-1**

* This script can be executed like any other shell command after associating the execute permission.
* The following command shows the execution step for our **Example-1** file.

**./Example-1**

**Running it as an argument of bash :**

This approach of script execution can be done to specify its name as the argument of the bashcommand.

* For example, the script named as **Example-1** can be executed by the use of following

command.

**bash Example-1**

* **Basic Shell Commands**
* **read**

The command read reads one line from the standard input (or from a file) and assign the word to the variable.

**Example**

read var\_year   
echo "The year is: $var\_year"

echo -n "Enter your name and press [ENTER]: "   
read var\_name  
echo "Your name is: $var\_name"

* **echo**

echo command in the bash shell writes its arguments to standard output**.**

The syntax for echo is

echo [option(s)] [string(s)]

The items in square brackets are optional. A *string* is any finite sequence of *characters* (i.e., letters, numerals, symbols and punctuation marks).

When used without any options or strings, echo returns a blank line on the display screen followed by the command prompt on the subsequent line.

* **Control Structures in shell**
* The bash shell provides many options to control the flow of script executions, typically known as control structures. The Table below list out these structures.

|  |  |
| --- | --- |
| **if** | If statement is used to execute commands based on certain conditions are met. It can be customized by the use of else to indicate what should happen if the condition does not satisfies. |
| **case** | This is used to handle multiple options in the script. It is same like switch statements used in C programming language. |
| **for** | This is a loop statement. The *for* statement is used to run a command for a given number of items. |
| **while** | Use while statement to execute the included statements as long as the specified condition is met. |
| **until** | This statement works exactly opposite of while statement. This can be used to execute a command until a certain condition is met. |

**If...Then...Else :**

* The if…then…else construct is used for flow control facility in shell scripting. This is usually used in conjunction with test command.
* This can be used for various activities like to find out the existence of a file, whether a variable currently has a value etc. The basic construction of if…then…else… statement is shown in below program code :

**General description of if…then construct**

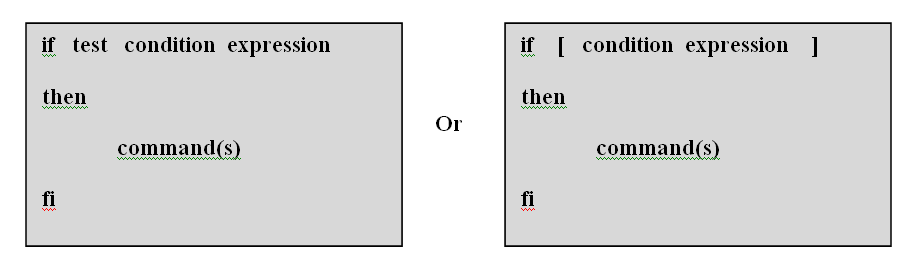
if condition expression

then

command(s)

fi

* This means one can use it to check one specific condition, and in case of true the command or group of commands associated will be executed.
* The if…then…fi construct can be extended by including else statement to handle all cases where the condition was not met. The Fig. below shows the actual syntax of the construct along with test command.



**Fig. If…then…fi statement**

* **System Calls in UNIX**
* The System Calls of an OS is the list of services or functions that the Operating System provides.
* System Calls run in a special mode in the CPU called Kernel Mode that uses an extended set of instructions and can access all the registers of the CPU.
* In contrast, application programs such as your web browser or your favorite editor run in User Mode
* **System Calls in user mode**
* **System Calls using C program**
* **System Calls for I/O**
* int open(const char \*path, int flags [ , int mode ] );
* int close(int fd);
* ssize\_t read(int fd, void \*buf, size\_t count);
* ssize\_t write(int fd, const void \*buf, size\_t count);
* off\_t lseek(int fd, off\_t offset, int whence);
* **Pseudo Code explanation for more understanding of the program:**
* fcntl.h -The header in the C POSIX library for the C programming language that **contains constructs that refer to file control**, e.g. opening a file, retrieving and changing the permissions of file, locking a file for edit, etc.
* The first parameter, argc (argument count) is an integer that indicates how many arguments were entered on the command line when the program was started. The second parameter, argv (argument vector), is an array of pointers to arrays of character objects.
* argv[0] points to the character string copy (the program name is conventionally the 0th parameter)
* argv[1] points to the character string oldfile, and argv[2]points to the character string newfile.
* The permission modes on the newly created file will be 0666 (octal), allowing all users access to the file for reading and writing.
* The open and create system calls return an integer called a file descriptor, which the program uses for subsequent references to the files.
* **INPUT: please specify input test file to be copied**
* **OUTPUT: please specify the expected output copied file for the input file**
* **FAQS:**

1. How does the Copy (CP) Command work and explain various options supported by copy (CP) Command?

#### Explain working of System Calls for I/O- Read, Write, Open, Close, lseek?

#### Explain working principle of ‘cp -p’ (Copy Command with -p option) command?

* **CONCLUSION**

The Linux shell provides several commands and programming language constructs, we have implemented Copy Command in ‘C’ programming language.