© 2014 IJIRT | Volume 1 Issue 11 | ISSN: 2349-6002

SHELL SCRIPTING AND SHELL PROGRAMMING IN UNIX

Shivangi Shandilya, Surekha Sangwan, Ritu Yadav
Dept. of Computer Science Engineering
Dronacharya College Of Engineering, Gurgaon

Abstract- **In this research paper, the concept of shell scripting and programming is discussed and various aspects of shell programming are also studied. A shell script is a computer program designed to be run by the UNIX shell which is a command line interpreter. The various dialects of shell scripts are considered to be scripting languages. Typical operations performed by shell scripts include file manipulation, program execution, and printing text. A shell script can provide a convenient variation of a system command where special environment settings, command options, or post-processing apply automatically, but in a way that allows the new script to still act as a fully normal UNIX command. The major concepts like Programming in the Borne and C-shell, in which it would be explained that how shell programming can be done in Borne and C-shell**.

I. INTRODUCTION

A shell script is a computer program designed to be run by the unix shell, a command line interpreter. The various dialects of shell scripts are considered to be scripting languages. Typical operations performed by shell scripts include file manipulation, program execution, and printing text. Users direct the operation of the computer by entering commands as text for a command line interpreter to execute, or by creating text scripts of one or more such commands. Users typically interact with a Unix shell using a terminal emulator, however, direct operation via serial hardware connections, or networking session, are common for server systems. The most influential Unix shells have been the Bourne shell and the C shell. These shells have both been used as the coding base and model for many derivative and work-alike shells with extended feature sets. The Bourne shell, sh, was written by Stephen Bourne at AT&T as the original Unix command line interpreter; it introduced the basic features common to all the Unix shells, including piping, here documents, command substitution, variables, control structures for condition-testing and looping and filename wildcarding. The language, including the use of a reversed keyword to mark the end of a block, was influenced by ALGOL 68.The C shell, csh, was written by Bill Joy while a graduate student at University of California, Berkeley. The language, including the control structures and the expression grammar, was modeled on C. The most generic sense of the term shell means any program that users employ to type commands. A shell hides the details of the underlying operating system and manages the technical details of the operating system kernel interface, which is the lowest-level, or "inner-most" component of most operating systems. In Unix-like operating systems, users typically have many choices of command-line interpreters for interactive sessions. When a user logs in to the system interactively, a shell program is automatically executed for the duration of the session. The type of shell, which may be customized for each user, is typically stored in the user's profile, for example in the local passwd file or in a distributed configuration system such as NIS or LDAP; however, the user may execute any other available shell interactively. The Unix shell is both an interactive command language as well as a scripting programming language, and is used by the operating system as the facility to control (shell script) the execution of the system.[2] Shells created for other operating systems often provide similar functionality. On hosts with a windowing system, like OS X, some users may never use the shell directly. On Unix systems, the shell has historically been the implementation language of system startup scripts, including the program that starts a windowing system, configures networking, and many other essential functions. However, some system vendors have replaced the traditional shell-based startup system (init) with different approaches, such as system. Graphical user interfaces for Unix, such as GNOME, KDE, and Xfce are sometimes called visual or graphical shells.

II. PROGRAM EXECUTION IN SHELL PROGRAMMING

A UNIX shell is a command language interpreter, the primary purpose of which is to translate command lines typed at a terminal into system actions. The shell itself is a program through which other programs are invoked. Although there are several different UNIX shells, among them the C shell, the Bourne shell and the Korn shell, the one most frequently used on Blake within Berkeley UNIX is the C shell. The shell has some built-in functions, which it performs directly, but most commands that you enter cause the shell to execute programs that are external to the shell. This sets the shell apart from other command interpreters, because it is just another user program at the same time that it functions almost exclusively as a mechanism for invoking other programs. The UNIX shell program interprets user commands, which are either directly entered by the user, or which can be read from a file called the shell script or shell program. Shell scripts are interpreted, not compiled. The shell reads commands from the script line per line and searches for those commands on the system, while a compiler converts a program into machine readable form, an executable file - which may then be used in a shell script. Apart from passing commands to the kernel, the main task of a shell is providing a user environment, which can be configured individually using shell resource configuration files.

III. PROGRAMMING IN BOURNE AND C SHEL

The Bourne shell was one of the major shells used in early versions of the Unix operating system and became a de facto standard. It was written by Stephen R. Bourne at Bell Labs and was first distributed with Version 7 Unix, circa 1977. Every Unix-like system has at least one shell compatible with the Bourne shell. The Bourne shell program name issh and its path in the Unix file system hierarchy is typically /bin/sh. On many systems, however, this may be a symbolic link or hard link to a compatible, but more feature-richshell than the Bourne shell. The POSIX standard specifies its standard shell as a strict subset of the Korn shell, an enhanced version of the Bourne shell. From a user's perspective the Bourne shell was immediately recognized when active by its characteristic default command line prompt character, the dollar sign ($).

Example of Bourne shell programming is given below:

To read commands from the terminal and process them:

```
#! /bin/sh
# usage: process sub-directory
dir=`pwd`
for i in *
do
if test -d $dir/$i
then
cd $dir/$i
while echo "$i:"
read x
do
eval $x
done
cd ..
fi
done
```

Process Sub-Directory This script will read and process commands in the named sub-directory. The user is prompted to supply the name of the command to be read in. This command is executed using the the built-in eval function.

The C shell was developed by Bill Joy for the Berkeley Software Distribution (BSD), a line of Unix operating systems derived from Unix and developed at the University of California, Berkeley. It was originally derived from the 6th Edition Unix shell (Thompson shell), with its syntax modeled after the C programming language. The C shell is used primarily for interactive terminal use, and less frequently for scripting and operating system control. It has interactive keyboard shortcuts in form of special control-key sequences for special effects such as job control.

The C shell is a command processor typically run in a text window, allowing the user to type commands. The C shell can also read commands from a file, called a script. Like all Unix shells, it supports filename wildcarding, piping, documents, command, variables and control structures for condition-testing and iteration. What differentiated the C shell from

others, especially in the 1980s, were its interactive features and overall style. Its new features made it easier and faster to use. The overall style of the language looked more like C and was seen as more readable.
Example of C shell programming is given below:

```
#!/bin/csh
If($days> 365) then
Echo This is over a year.
Endif
```

IV. ADVANTAGES OF SHELL PROGRAMMING

A. Easy to use.
B. Quick start and interactive debugging.
C. Time saving.
D. System admin task automation.
E. Shell scripts can execute without any additional effort on nearly any modern UNIX / Linux / BSD / Mac OS X operating system as they are written an interpreted language.
F. To automate the frequently performed operations
G. To run sequence of commands as a single command.
H. Portable (It can be executed in any Unix-like operating systems without any modifications)

V. DISADVANTAGES OF SHELL PROGRAMMING

A. Compatibility problem between different platforms.
B. Slow execution speed.
C. A new process launched for almost every shell command executed.

REFERENCES

[1] http://linuxshellscripting.blogspot.in/2011/02/shell-scripting-advantages-and.html
[2] http://www.thegeekstuff.com/2010/07/execute-shell-script/
[3] http://en.wikipedia.org/wiki/Shell_script
[4] http://www.calpoly.edu/~rasplund/script.html
[5] http://supportweb.cs.bham.ac.uk/docs/tutorials/docsystem/build/tutorials/unixscripting/unixscripting.html