Lecture - 19

References : Sumitabha Das

SECTION -D

SHELL PROGRAMMING

References : Sumitabha Das

INTRODUCTION

- C Shell Programming
- Shell Programming Practices

C Shell Programming

- The C shell provides a rich scripting language that, at best, has a slight similarity to the programming language C.
- Shell scripting languages provide the user with a great many tools for handling everyday tasks around the system and even some less everyday tasks.
- A shell script can contain Unix commands as well as the shell commands
- Unlike compiler based languages, shell scripts are executed by the shell one line at a time. While this will obviously make for slower performance, advantage is gained in the ease of modifying programs without all of the hassle of compiling and linking.
- All that is required for a shell script to be executed is that it be made executable with the following command: **$ chmod u+x script_name**

- Shell scripts are often written to handle some of the more tedious tasks that a user encounters on a regular basis. A simple C shell script could be a list of Unix commands that archives and compresses the users home directory and copies it to a specified mounted disk partition for storage

```
#!/bin/csh
# backup tars and compresses ~/ and puts in storage on /dsk2/strg/
#
 tar -cvf dec18_95.nbdat.tar ~/
compress dec18_95.nbdat.tar
 cp dec18_95.nbdat.tar.Z /dsk2/strg/
```

- With the exception of the lines that start with hash marks (#), the script is a list of simple Unix commands.
- This task could most certainly have been entered on a single command line with use of a pipeline, but it illustrates the basic format of a C shell script.
- Almost any line starting with a hash mark will be ignored by the shell and hence indicate programmer comments.
- The one exception to this rule is the hash bang (#!) sequence of characters, this has special meaning to the shell.
- It tells the shell which environment to start for execution of the script. This could be any shell or even other scripting environments such as perl (Practical Extraction and Report Language) ,or tcl (Tool Command Language) ,which are Unix scripting languages but not shells (at least not interactive shells like those discussed in this book).

- The shells are usually found in the /bin directory, but this might differ from system to system. The powerful feature of shell scripts over simply writing the commands on a command line is that scripts can contain many types of safety, logging, and other features to provide a worry free and organized working environment. As the scripts in this chapter begin to become more complex, this point should become clear.

PROGRAM 1

WRITE A PROGRAM TO ADD TWO NUMBERS

```
echo enter 1
read a
echo enter 2
read b
c=`expr $a + $b`
echo addition = $c
```

Output

```
enter 1
7
enter 2
3
addition =10
```

WRITE A PROGRAM TO FIND LARGEST OF THREE NUMBERS

```
echo enter 1
read n1
echo enter 2
read n2
echo enter 3
read n3

if [ $n1 -gt $n2 ]&&[ $n1 -gt $n3 ]
 then          echo $a is big
elif [ $n2 -gt $n3 ]&&[ $n2 -gt $n3 ]
 then         echo $b is big
 else         echo $c is big
fi
```

Output

```
enter 1
4
enter 2
3
enter 3
2
4 is big
```

WRITE A PROGRAM TO FIND CURRENT DATE AND DIRECTORY

```
echo current date=`date`
echo user =`who am i`
echo current dir =`pwd`
```

Output

```
current date =Fri Nov 16 13:12:25 IST 2007
user =localhost.localdomain!user5 pts/1 Nov 16 12:48 (192.168.1.46)
current dir =/home/user5
```

WRITE A SHELL PROGRAM TO PERFORM OPERATIONS USING CASE STATEMENT AS
A)ADDITION
B)SUBSTRACTION
C)MULTIPLICATION
D)DIVISION

```
echo  a b
read a b

echo a= add
echo b= sub
echo c= mul
echo d= div

echo ch
 read ch

case $ch  in
a)
let z= $a + $b
echo add= $z
 ;;
```

```
b)
 let z= $a - $b
echo sub=$z
 ;;

c)
let z= $a * $b
echo mul= $z
 ;;
```

```
d)
let z= $a / $b
echo div= $z
;;

*)
echo invalid option
;;
Esac
```

Output

```
a b
3 4

a= add
 b= sub
 c= mul
 d= div

ch
a
add = 7
```

- Refer shell programs from your POS Lab Practical file.. They are important

APPLICATIONS

- System boot scripts (/etc/init.d)
- System administrators, for automating many aspects of computer maintenance, user account creation etc.
- Application package installation tools
- Application startup scripts, especially unattended applications (e.g. started from **cron** or **at**)
- Any user needing to automate the process of setting up and running commercial applications, or their own code.

RESEARCH

- DUBLIN--(BUSINESS WIRE)--**Research and Markets** (http://www.researchandmarkets.com/research/a59aa9/shell_scripting_e) has announced the addition of John Wiley and Sons Ltd's new book "Shell Scripting: Expert Recipes for Linux, Bash and more" to their offering.
- “Shell Scripting: Expert Recipes for Linux, Bash and more”
- A compendium of shell scripting recipes that can immediately be used, adjusted, and applied
- The shell is the primary way of communicating with the Unix and Linux systems, providing a direct way to program by automating simple-to-intermediate tasks. With this book, Linux expert Steve Parker shares a collection of shell scripting recipes that can be used as is or easily modified for a variety of environments or situations. The book covers shell programming, with a focus on Linux and the Bash shell; it provides credible, real-world relevance, as well as providing the flexible tools to get started immediately.
- Shares a collection of helpful shell scripting recipes that can immediately be used for various of real-world challenges
- Features recipes for system tools, shell features, and systems administration
- Provides a host of plug and play recipes for to immediately apply and easily modify so the wheel doesn't have to be reinvented with each challenge faced
- Come out of your shell and dive into this collection of tried and tested shell scripting recipes that you can start using right away!