Essential Shell Programming

Part IV

Session Objective

We will be learning the following constructs

- While varieties
- changing basenames
- set and shift
- trap

While: Looping Varieties

Wait for a file

```
#! /bin/sh
While [! –r invoice.lst] do
Sleep 60
done
Echo “file created”
```

While: Looping Varieties

Infinite loop

```
While true ; do
df –t
Sleep 300
done &
```

basename: Changing Filename Extensions

- They are useful in chaining the extension of group of files.
- Basename extracts the base filename from an absolute pathname.

Example1: $basename /home/user1/test.pl

Output: test.pl

Example2: $basename test2.doc doc

Output: test2

set and shift: Manipulating the Positional Parameters

The set statement assigns positional parameters $1, $2 and so on, to its arguments. This is used for picking up individual fields from the output of a program.

set and shift: Manipulating the Positional Parameters

Example 1: $ set 9876 2345 6213

This assigns the value 9876 to the positional parameters $1, 2345 to $2 and 6213 to $3

Shift: Shifting Arguments Left

Shift transfers the contents of positional parameters to its immediate lower numbered one.

Example 1:

```
$ echo “$@”
```

$@ and $ are interchangeable*

```
Mon Oct 8 08:02:45 IST 2007
$ echo $1 $2 $3
Mon Oct 8
```

Shift: Shifting Arguments Left

```
$ shift
$ echo $1 $2 $3
Oct 8 08:02:45
$ shift 2
08:02:45 IST 2007
```

Shift: Shifting Arguments Left

```
#!/bin/sh
Case $# in
0|1) exit 2;;
   *) $fname = $1
      shift
      for pattern in “$@”; do
        grep “$pattern” $fname || echo “$pattern
         not found”
    done;; esac
```

Set -- : Helps Command Substitution

- In order for the set to interpret - and null output produced by UNIX commands the – option is used .
- If not used – in the output is treated as an option and set will interpret it wrongly. In case of null, all variables are displayed instead of null.

Contd..

Example1: $set `ls –l chp1`

Output: -rwxr-xr-x: bad options

Example2: $set `grep usr1 /etc/passwd`

Correction to be made to get correct output are:

```
$set -- `ls –l chp1`
$set -- `grep usr1 /etc/passwd`
```

The Here Document (<<)

- The shell uses the << symbol to read data from the same file containing the script.
  - Here document.
- Signifying that the data is here rather than in a separate file.
- Any command using standard input can also take input from a here document.

The Here Document (<<)

Example:

```
mailx kumar << MARK
Your program for printing the invoices has
   been executed
on `date`.Check the print queue
The updated file is $flname
MARK
```

The Here Document (<<)

Using Here Document with Interactive Programs:

A shell script can be made to work non-interactively by supplying inputs through here document. Example:

```
$ emp.sh << END
> director
>emp.lst
>END
```

The Here Document (<<)

Output:

```
Enter the pattern to be searched: Enter the file
   to be used: Searching for director from file
   emp.lst
9876 Jai sharma Director Productions
2356 Rohit            Director Sales
Selected records shown above.
```

trap: interrupting a Program

- shell scripts terminate - interrupt key is pressed.
- lot of temporary files will be stored on disk.
- The trap statement lets you do the things you want to do when a script receives a signal.
- trap placed at the beginning of the shell script

uses two lists:

trap ‘command_list’ signal_list

trap: interrupting a Program

When a script is sent any of the signals in signal_list, trap executes the commands in command_list.

The signal list can contain the integer values or names (without SIG prefix) of one or more signals – the ones used with the kill command.

Example: To remove all temporary files named after the PID number of the shell:

trap: interrupting a Program

```
trap ‘rm $$* ; echo “Program Interrupted” ; exit’
   HUP  INT  TERM
```

trap is a signal handler.

Removes all files expanded from $$*, echoes a message and finally terminates the script when signals SIGHUP (1), SIGINT (2) or SIGTERM(15) are sent to the shell process running the script.

trap: interrupting a Program

A script can also be made to ignore the signals by using a null command list.

Example:

```
trap ‘’ 1  2  15
```

Debugging shell scripts with set -x

```
$ emp.sh emp.lst 22 12 01
+flname=emp.lst
+shift
+grep 22 emp.lst
22 shekar gm  sales   12/08/07
+grep 12 emp.lst
12 xyz    director  marketing  15/10/07
```

Sample scripts

```
#!/bin/sh
IFS="|"
While echo "enter dept code:\c"; do
Read dcode
Set -- `grep "^$dcode"<<limit
01|ISE|22
02|CSE|45
03|ECE|25
04|TCE|58
limit`
```

Sample scripts

```
Case $# in
3) echo “dept name :$2 \n  emp-id:$3\n”
*) echo “invalid code”;continue
esac
done
```

Sample scripts

Output:

```
$valcode.sh
Enter dept code:88
Invalid code
Enter dept code:02
Dept name  : CSE
Emp-id :45
Enter dept code:<ctrl-c>
```

Sample scripts

```
#!/bin/sh
x=1
While [$x –le 10];do
 echo “$x”
 x=`expr $x+1`
done
```

Sample scripts

```
#!/bin/sh
sum=0
for I in “$@” do
 echo “$I”
sum=`expr $sum + $I`
done
Echo “sum is $sum”
```

Sample scripts

```
#!/bin/sh
sum=0
for I in `cat list`; do
 echo “string is $I”
x= `expr “$I”:’.*’`
Echo “length is $x”
done
```

Conclusion

In this session we saw how positinal parameters can be made use of in different varieties of ways.

Additionally learnt about some signaling mechanism as well as varieties in using loop constructs

Concluding Essential Shell Programming

Learnt

✓ Shell program/shell script

✓ writing scripts to take on information interactively as well as from command line

✓ termination status of program

✓ combining commands using logical operators

Concluding Essential Shell Programming

- ✓ Decision making using if,case,while & for
- ✓ Testing expressions
- ✓ Evaluation of Expressions
- ✓ Script by different names
- ✓ Setting & manipulating positional parameters
- ✓ Signal handling using trap

Concluding Essential Shell Programming

✓ Powerful scripts can be built using the excellent features offered in shell programming. A preliminary insight on shell programming has been offered here.

✓Scripts can make use of these as well as advanced features to help programmers/users of system to work with efficiency and ease.

THANK YOU