Shell Programming

Using Command Line Arguments

Shell parameter	Significance
$1, $2...	Positional parameters representing command line arguments
$ #	No. of arguments specified in command line
$ 0	Name of the executed command
$ *	Complete set of positional parameters as a single string
"$ @"	Each quoted string treated as separate argument
$?	Exit status of last command
$$	Pid of the current shell
$!	PID of the last background job.

exit and Exit Status of Command

- To terminate a program exit is used.
- Nonzero value indicates an error condition.
  Example 1:
  $ cat foo
  Cat: can’t open foo
- Returns nonzero exit status.
- The shell variable $? Stores this status.

The logical Operators && and ||

- Two operators that allow conditional execution, the && and ||.

  Usage:
  cmd1 && cmd2
  cmd1 || cmd2

- && delimits two commands. cmd 2 executed only when cmd1 succeeds.

- Example:
```
if [ "$data2" != "world" && "$data1" != "hello"]
then
{
echo "good afternnon
}
```

- The right side of && will only be evaluated if the exit status of the left side is zero.
- || is the opposite: it will evaluate the right side only if the left side exit status is nonzero

- $ false && echo hello
- $ true && echo hello
  hello
- $ true || echo hello
- $ false || echo hello
  hello

Using test and [ ] to Evaluate Expressions

- Test works in three ways:

- Compare two numbers
- Compares two strings or a single one for a null value
- Checks files attributes

- Test doesn’t display any output but simply returns a value that sets the parameters $?

Using test and [ ] to Evaluate Expressions

- **String Comparison**

Test	True if
s1=s2	String s1=s2
s1!=s2	String s1 is not equal to s2
-n stg	String stg is not a null string
-z stg	String stg is a null string
stg	String stg is assigned and not null
s1==s2	String s1=s2

```
#!/bin/bash
echo "Enter a string "
read s1
if [ -z "$s1" ]; then
echo "Entered  string is null "
fi
```

- *if [! -n "$s2"] #?*
- *if test "$s1" == "$s2" #?*

-a(AND), -o(OR) operators

- Example:

```
if [ -n "$s1" -a -n "$s2" ]; then
echo "strings are non null"
fi
```

File Tests

- test can be used to test various file attributes like its type (file, directory or symbolic links)
- its permission (read, write. Execute, SUID, etc).

- Example:

```
$ ls –l emp.lst
-rw-rw-rw- 1 kumar group      870 jun 8 15:52
   emp.lst
$ [ -f emp.lst ] ; echo $?
0                                  → Ordinary file
```

File Tests

- $ [-x emp.lst] ; echo $? → Not an executable.
  1
- $ [! -w emp.lst] || echo “False that file is not writeable”

  False that file is not writable.

File Tests

Test	True if
-f file	File exists and is a regular file
-r file	File exists and readable
-w file	File exists and is writable
-x file	File exists and is executable
-d file	File exists and is a directory
-s file	File exists and has a size greater than zero
-e file	File exists (Korn & Bash Only)
-u file	File exists and has SUID bit set
-k file	File exists and has sticky bit set
-L file	File exists and is a symbolic link (Korn & Bash Only)
f1 –nt f2	File f1 is newer than f2 (Korn & Bash Only)
f1 –ot f2	File f1 is older than f2 (Korn & Bash Only)
f1 –ef f2	File f1 is linked to f2 (Korn & Bash Only)

The case Conditional

- Syntax:
- case expression in
- Pattern1) commands1 ;;
- Pattern2) commands2 ;;
- Pattern3) commands3 ;;
- …
- esac

The case Conditional

- Example:

```
read choice
case “$choice” in
    1) ls –l;;
    2) ps –f ;;
    3) date ;;
    4) who ;;
    5) exit ;;
    *) echo “Invalid option”
esac
```

The case Conditional

- **Matching Multiple Patterns:**
- case can also specify the same action for more than one pattern .
- For instance to test a user response for both y and Y (or n and N).
- Example:

```
Read ans
Case “$ans” in
Y | y );;
N | n) exit ;;
esac
```

expr: Computation and String Handling

- The Bourne shell uses expr command to perform computations and string manipulation.

- expr can perform the four basic arithmetic operations (+, -, *, /), as well as modulus (%) functions.
- **Computation:**
  Example1 :$ x=3 y=5
  $ expr $x+$y
  8 → Output
  Example 2:$ expr 3 * 5
  15 → Output

- *Note: \ is used to prevent the shell from interpreting * as metacharacter*