Essential Shell Programming

Part II

What we will be learning

- exit
- Logical operators
- test
- if

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.

exit and Exit Status of Command

Example 2:

```
grep director emp.lst > /dev/null:echo $?
0
```

Exit status is used to devise program logic that branches into different paths depending on success or failure of a command

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.

The logical Operators && and ||

Example1:

```
$ grep  ‘director’ emp.lst && echo “Pattern found”
```

Output:

```
9876  Jai Sharma Director Productions
2356  Rohit           Director Sales
Pattern  found
```

The logical Operators && and ||

Example 2:

```
$ grep  'clerk' emp.lst || echo "Pattern not
  found"
```

Output:

```
Pattern not found
```

Example 3:

```
grep "$1" $2 || exit 2
echo "Pattern Found Job Over"
```

The if Conditional

Form 1

```
if command is successful
then
 execute commands
fi
```

Form 2

```
if command is successful
then
execute commands
else
 execute commands
fi
```

Form 3

```
if command is successful
then
execute commands
elif command is successful
then...
 else...
 fi
```

If the command succeeds, the statements within if are executed or else statements in else block are executed (if else is present).

The if Conditional

Example:

```
#! /bin/sh
  if grep “^$1” /etc/passwd 2>/dev/null
# ^ matching at the beginning of the line
  then
      echo “Pattern Found”
  else
      echo “Pattern Not Found”
  fi
```

The if Conditional

Output1:

```
$ emp3.sh ftp
ftp: *.325:15:FTP
   User:/Users1/home/ftp:/bin/true
Pattern Found
```

Output2:

```
$ emp3.sh mail
Pattern Not Found
```

Using test and [ ] to Evaluate Expressions

- Test statement is used to handle the true or false value returned by expressions.
- Test uses certain operators to evaluate the condition on its right
- Returns either a true or false exit status
- Is used by if for making decisions.

Using test and [ ] to Evaluate Expressions

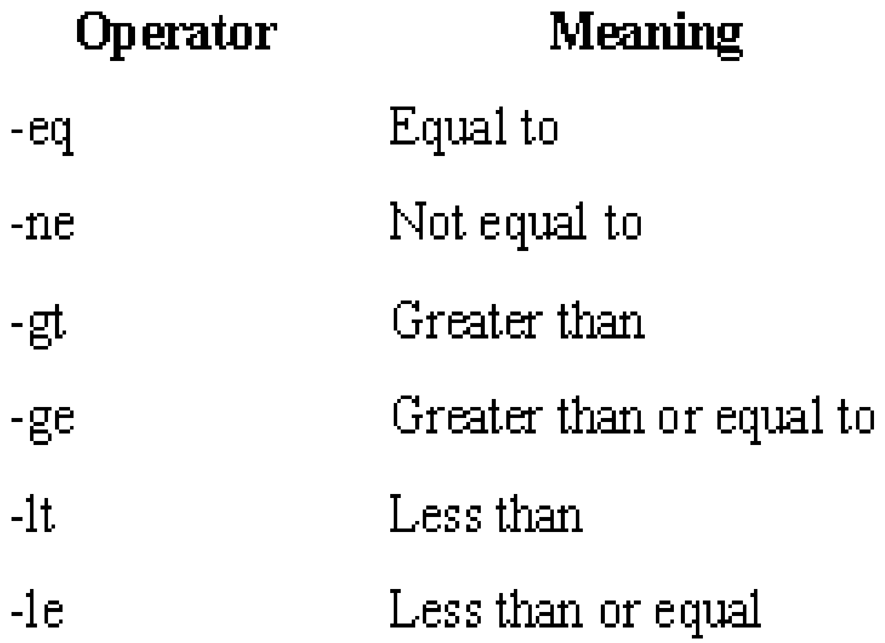
Test works in three ways:

- 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

Numeric Comparison:



Operator	Meaning
-eq	Equal to
-ne	Not equal to
-gt	Greater than
-ge	Greater than or equal to
-lt	Less than
-le	Less than or equal

Using test and [ ] to Evaluate Expressions

Numeric Comparison:

Ex:

```
$ x=5;y=7;z=7.2
1. $ test $x –eq $y; echo $?
     1                    Not equal
2. $ test $x –lt $y; echo $?
    0                     True
```

Using test and [ ] to Evaluate Expressions

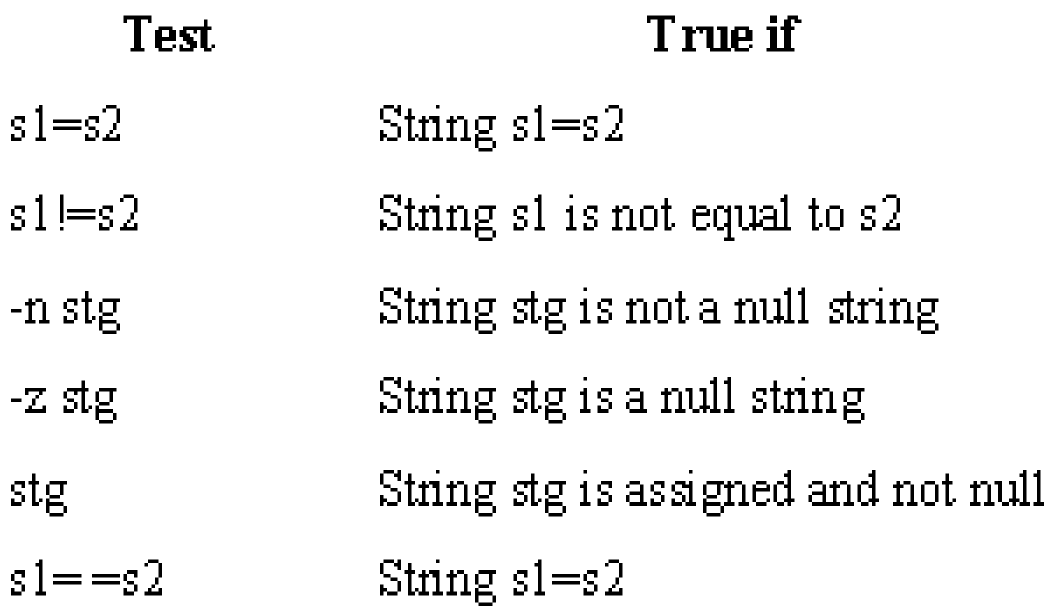
Shorthand for test

[and] can be used instead of test. The following two forms are equivalent

```
Test $x –eq $y
      and
[ $x –eq $y ]
```

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

Using test and [ ] to Evaluate Expressions

Example:

```
#!/bin/sh
#emp1.sh checks user input for null values
   finally turns emp.sh developed previously
#
if [ $# -eq 0 ] ; then
echo “Enter the string to be searched :\c”
read pname
```

Using test and [ ] to Evaluate Expressions

```
if [ -z “$pname” ] ; then
echo “You have not entered the string”; exit 1
fi
echo “Enter the filename to be used :\c”
read flname
if [ ! –n “$flname” ] ; then
echo “ You have not entered the flname” ; exit
   2
 fi
```

Using test and [ ] to Evaluate Expressions

```
grep "$pname" $flname
fi

Output1: $emp1.sh
Enter the string to be searched :[Enter]
You have not entered the string
```

Using test and [ ] to Evaluate Expressions

```
Output2:  $emp1.sh
Enter the string to be  searched :root
Enter the filename to be searched :/etc/passwd
Root:x:0:1:Super-user:/:/usr/bin/bash
```

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
```

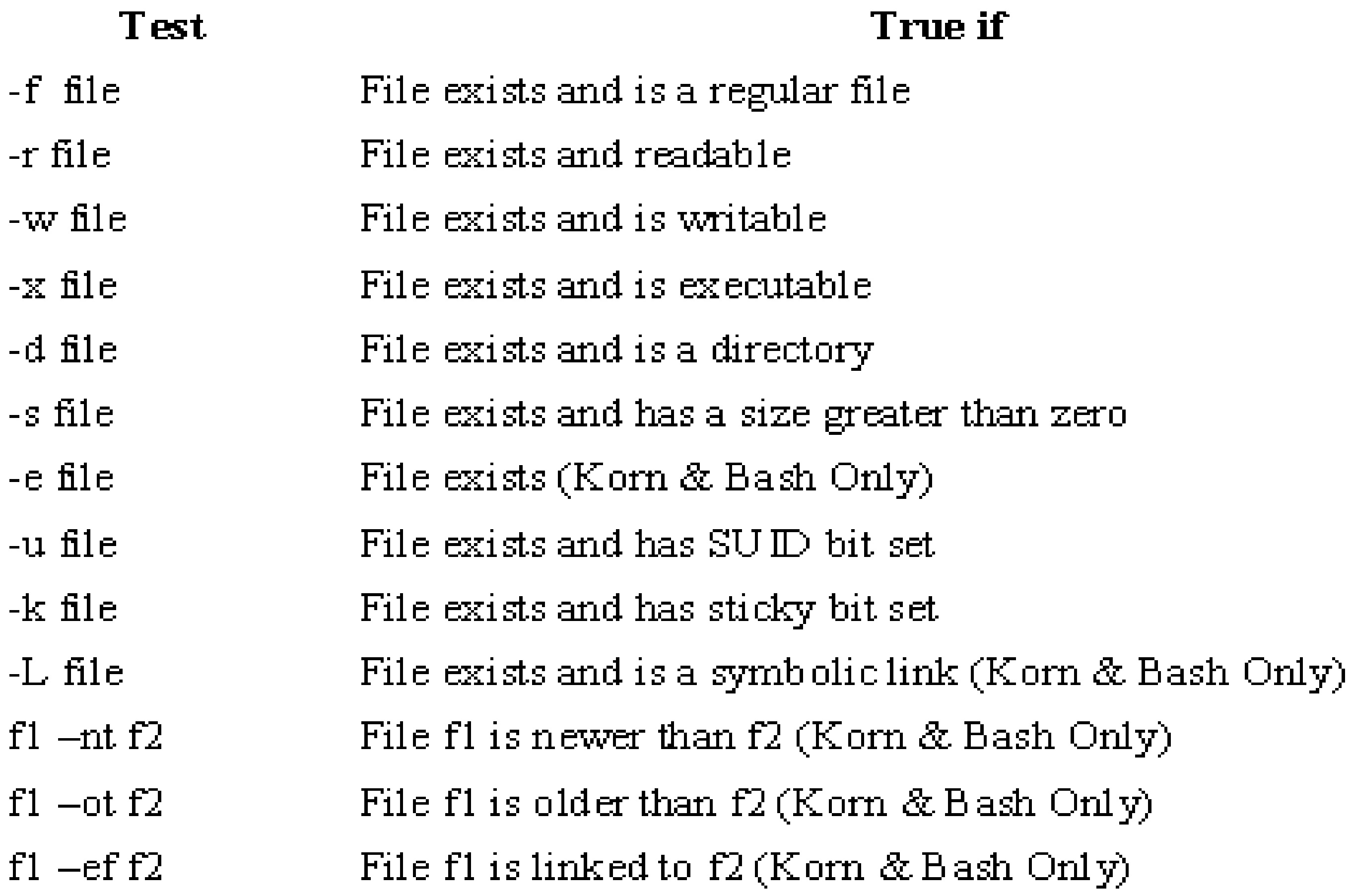
File Tests

```
$ [-x emp.lst] ; echo $?
1
```

→ Not an executable.

```
$ [ -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)

Conclusion

In this session we have learnt

- Termination status of a program
- Command being combined using logical operators
- Numeric, string and file test operations using test
- Decision making structure if