

UNIX

Lesson 16: Programming Construct

Lesson Objectives



- At the end of the session you will be able to understand:
 - Conditional execution
 - Programming constructs



16.1: Conditional Execution Details



Logical Operators && and ||:

- && operator delimits two commands. Second command is executed only if the first succeeds.
- | operator delimits two commands. Second command is executed only if the first fails.
- Example:

\$grep `director` emp.lst && echo "pattern found"
\$grep `manager` emp.lst || echo "pattern not found"

16.2: if else Details



```
Syntax
(i) if <condition is true>
   then
       <execute commands>
   else
       <execute commands>
fi
(ii) if <condition is true>
     then
     <execute commands>
     fi
```

```
Example
if grep "^$1" /etc/passwd 2>/dev/null
 then
         echo "pattern found"
 else
         echo "pattern not found"
fi
```

if Statement

fi



```
Syntax:
 (iii) if <condition is true>
     then
      <execute commands>
     elif < condition is true>
     then
      <execute commands>
      <...>
  else
   <execute commands>
```

Example

```
if test $# -eq o; then
 echo "wrong usage " > /dev/tty
 elif test $# -eq 2; then
   grep "$1" $2 || echo "$1 not
         found in $2" > /dev/tty
  else
     echo "you didn't enter 2
                  arguments"
fi
```

Relational Operator for numbers



Specify condition either using *test* or [condition]

• Example: test \$1 -eq \$2 same as [\$1 -eq \$2]
Relational Operator for Numbers:

• eq: Equal to

• ne: Not equal to

gt: Greater than

gc: Greater than or equal to

It: Less than

Ic: Less than or equal to



Relational Operator for strings and logical operators

String operators used by *test:*

-n strTrue, if str not a null string

-z strTrue, if str is a null string

• S1 = S2 True, if S1 = S2

■ S1 != S2 True, if S1 \neq S2

str True, if str is assigned and not null

Logical Operators

- -a .AND.
- •-o .OR.
- •! Not

File related operators



File related operators used by test command

-f <file></file>	True, if file exists and it is regular file
-d <file></file>	True, if file exist and it is directory file
-r <file></file>	True, if file exist and it is readable file
-w <file></file>	True, if file exist and it is writable file
-x <file></file>	True, if file exist and it is executable file

-s <file> True, if file exist and it's size > 0

-e <file> True, if file exist

Example



Check whether user has entered a filename or not:

• Example:

```
echo "Enter File Name:\c "
read fn
if [ -z "$fn" ]
then
echo "You have not entered file name"
fi
```

16.3:test operator Example



Example:

if test
$$x - eq$$

 $\equiv if [x - eq$

Example:

Example



```
echo "Enter the source file name:\c"
read source
#check for the existence of the source file
if test -s "$source" #file exists & size is > 0
then
  if test! -r "$source"
  then
         echo "Source file is not readable"
         exit
  fi
else
  echo "Source file not present"
  exit
fi
```

Case command



```
Syntax:
      case <expression> in
      <pattern 1> ) <execute</pre>
      commands>;;
      <pattern 2> ) <execute</pre>
      commands>;;
              <...>
              <...>
      esac
```

```
Example:
echo "\n Enter Option:\c"
 read choice
 case $choice in
 1) ls -l ;;
 2) ps -f ;;
 3) date ;;
 4) who ;;
 5) exit ;;
 esac
```

Example



```
echo "do you wish to continue?"
read ans
    Case "$ans" in
    [yY] [eE] [sS]) ;;
    [nN] [oO]) exit ;;
    *) "invalid option" ;;
esac
```





Syntax:

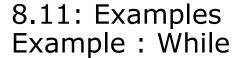
while <condition is true>

do

<execute statements>

done

```
e.g.
while [$x-gt3]
         do
                  ps-a
                  sleep 5
         done
while true
         do
                  ps -a
                  sleep 5
         done
```









continue:

- Suspends statement execution following it.Switches control to the top of loop for the next iteration.

break:

Causes control to break out of the loop.

exit:

• The exit statement is used to "terminate" a script that is running.

8.12: Break & Continue Statement Example



```
while echo "designation:\c"
do
          read desig
          case "$desig" in
          [0-9]) if grep "^$desig" emp.lst >/dev/null then
                     echo "start is desig"
                fi;;
          esac
done
```

8.13: until loop Syntax



Complement of *while* statement. Loop body executes repeatedly as long as the condition remains *false*. • Example:

until false
do
ps -a
sleep 5
done

8.14: For Statement for statement



```
Syntax:
   for variable in list
   do
           <execute
   commands>
   done
```

```
Eg:
for x in 1 2 3
    do
         echo "The value of x is $x"
    done
for var in $PATH $HOME $MAIL
    do
         echo "$var"
   done
for file in *.c
    do
         cc $file
    done
```

Example: for



```
for file in chap20 chap21 chap22 chap23;
do
cp $file ${file}.bak
echo $file copied to $file.bak
done
```

for file in 'cat clist'......

for file in *.htm *.html; do # do something done

for pattern in "\$@"; do grep "\$pattern" emp.lst || echo "\$pattern not found" done

8.14 For Statement Details



Syntax:

```
for (( expr1; expr2; expr3
))
do
..... repeat all
statements between
do and done until
expr2 is TRUE
```

done

e.g.

8.15: Examples Example: Until



```
#script to create a employee file
ans="y"
until [ $ans = "N" -o $ans = "n" ]
do
          echo "Enter the name:\c"
          read name
          echo "Enter the grade :\c"
          read grade
          echo "Enter the basic :\c"
          read basic
          echo $name: $grade: $basic >>emp
echo "Want to continue (Y/N):\c"
read ans
done
#end of script
```

SUMMARY

- In this lesson, you have learnt:
 - Implementing programming construct i.e. if ,for while, until and case
 - Use of logical || and &&

Review Questions

Question 1: What test operator return?



© 2018 Capgemini. All rights reserved.