Lecture 6

Shell Programming: Control constructs, loops

COP 3344 Introduction to UNIX

Fall 2007

Acknowledgment: These slides are modified versions of Prof. Sudhir Aggarwal's slides

Simple *if* Statement

- General form

```
if condition
then
  one-or more commands
fi
```

myrm

```
#!/bin/sh
if [ ! -f $1 ]
then
  echo $0: No file named $1
fi
if [ -f $1 ]
then
  rm $1
  echo Removed file: $1
fi
```

```
$ ./myrm file.txt
Removed file: file.txt
$./myrm file.txt
./myrm: No file named file.txt
```

Testing Conditions

- There are two ways to test for conditions

  `test condition`

  `[ condition ]`

  – A condition can be reversed with a `!`

Test File Attributes

- `[ -r file1 ]`
  – Is *file1* readable?
- `[ -w file1 ]`
  – Is *file1* is writeable?
- `[ -x file1 ]`
  – Is a *file1* is executable?
- `[ -f file1 ]`
  – Does *file1* exist?

Testing Numeric Values

- Use: `-eq`, `-ne`, `-gt`, `-ge`, `-lt`, or `-le`
- Examples
  – `[ $1 –lt $2 ]`
    - Is *$1* less than *$2*?
  – `[ $1 –gt 0 ]`
    - Is *$1* greater than *0*?

More Conditions

- Testing strings
  – It is a good idea to put the shell variable being tested inside double quotes

    `[ “$1” = “yes” ]`

  – Note
    - `[$1 != “yes” ]` becomes `[ != “no” ]` if *$1* is empty, and will lead to a syntax error
- Testing multiple conditions
  – Operators
    - `&&` is the *and* operator
    - `||` is the *or* operator
  – Examples

    `[ “$1” = “yes” ] && [ -r $2.txt ]`

    `[ “$1” = “no” ] || [ “$2” = “maybe” ]`

General *if* Statement

- General form

```
if condition
then
  commands
elif condition
then
  commands
…
else
  commands
fi
```

  – You can have *0* or more *elif* statements
  – The *else* is optional

myrm2

```
#!/bin/sh
if [ ! -f $1 ]
then
  echo $0: No file named $1
elif [ -f $1 ]
then
  rm $1
  echo Removed file: $1
fi
```

myrm3

```
#!/bin/sh
if [ -f $1 ]
then
  rm $1
  echo Removed file: $1
else
  echo $0: No file named $1
fi
```

Case Statement

- General form

```
case stringvalue in
pattern1) commands;;
pattern2) commands;;
…
      *) commands;;
esac
```

  – Compares *stringvalue* to each pattern
  – At a match, perform the corresponding commands.
  – The *;;* indicates that it should jump to the statement after the *esac*
  – The **)* gives the default case

myrm4

```
#!/bin/sh
echo “Do you really want to
delete $1? (yes/no)”
read choice
case “$choice” in
yes) rm $1;
     echo Deleted $1;;
no)  echo Did not delete $1;;
*)   echo Invalid choice;;
esac
```

```
$./myrm4 file1
Do you really want to
delete file1? (yes/no)
yes
Deleted file1
```

for statement

- General form
  ```
  for variable [ in word_list ]
  do
     commands
  done
  ```
  - The *commands* are executed several times
  - Each time, the *variable* is assigned a different word in the *word_list*
  - If *in word_list* is omitted, then *variable* is assigned each of the command line arguments

backup
```
#!/bin/sh
for filename
do
  cp $filename $filename.bak
done
```

```
$ls
myrm2
tests.sh
$./backup tests.sh myrm2
$ls
myrm2
myrm2.bak
tests.sh
tests.sh.bak
```

backup2
```
#!/bin/sh
list="myrm2 tests.sh"
for filename in $list
do
  cp $filename $filename.bak
done
```

while statement

- General form
  ```
  while condition
  do
    commands
  done
  ```

calc
```
#!/bin/sh

read var1 op var2
while [ $var1 != quit ]
do
  echo $var1 "$op" $var2 = `expr $var1 "$op" $var2`
  read var1 op var2
done
```

```
./calc
2 + 4
2 + 4 = 6
3 * 5
3 * 5 = 15
4 / 2
4 / 2 = 2
quit
```

Using *exit*

- The *exit* command makes the shell script terminate
  - It can set the *status* at the time of exit
- General form
  `exit` or `exit status`
  - Zero normally indicates success
  - Nonzero values indicate some type of failure

Example of *exit*

myrm5
```
#!/bin/sh
if [ -f $1 ]
then
  rm $1
  exit 0
else
  exit 1
fi
```

rmfiles
```
#!/bin/sh
for filename
do
if ./myrm5 $filename
then
  echo Removed file: $filename
else
  echo Unable to remove
$filename
fi
done
```

```
$ ./rmfiles asj ddaas
Unable to remove asj
Unable to remove ddaas
$ ./rmfiles *.bak
Removed file: calc.bak
Removed file: myrm.bak
```