Unix/Linux commands and shell programming-Part 2
(Dr. Mohamed El Bachir Menai)
Shell logic structures

- The basic logic structures needed for program development are:
  - Sequential structure
  - Decision structures
  - Looping structures
Sequential structure

- Sequential structure states that commands will be executed in the order in which they appear in the program.
- The only break in this sequence comes when a branch instruction changes the flow of execution.
Decision structures

- The if statement is the primary decision-making control structure

```plaintext
if (expression) simple-command
or
if (expression) then
...
[else
...
] endif
```
Decision structures, cont

- bash

```bash
if test1; then
commands1;
[elif test2; then
commands2;]
[else commands3;]
fi
```
Decision structures, cont.

Example

```csh
#!/bin/csh
if ($#argv != 2) then
    echo $0 needs two parameters!
    echo You are inputting $#argv parameters.
else
    set par1 = $argv[1]
    set par2 = $argv[2]
endif

The set built-in command can be used to display all variables set in the shell
```
Decision structures, cont.

- Another example:

```bash
#!/bin/csh
# number is positive, zero or negative
echo "enter a number:"
set number = $<
if ( $number < 0 ) then
    echo "negative"
else if ( $number == 0 ) then
    echo " zero "
else
    echo " positive "
endif
```
Decision structures, cont.

- The switch logic structure simplifies the selection of a match when you have a list of choices.
- It allows your program to perform one of many actions, depending upon the value of a variable.
Decision structures, cont.

```c
switch ( var )
    case string1:
        command_set_1
        breaksw
    case string2:
        command_set_2
        breaksw
    default
        command_set_3
endsw
```
Decision structures, cont.

#!/bin/csh
if ($#argv == 0 ) then
    echo "No arguments supplied...exiting"
else
    switch ($argv[1])
        case [yY]:
            echo " Argument one is yes. "
            break
        case [nN]:
            echo " Argument one is no. "
            break
        default:
            echo " Argument one is neither yes nor no. "
            break
    ends
endif
Decision structures, cont.

- bash
  case word in
    pattern {\texttt{|pattern}*} commands;;
  ...
  esac
Decision structures, cont.

- Example
  
  ```bash
  case $reply in
    "1")
      date;;
    "2"|"3")
      pwd;;
    *)
      echo illegal choice
    ;;
  esac
  ```
Looping structures

while ( expr )
    command_set
end

foreach var ( worddlist )
    command_set
end
Looping structures, cont.

- **bash**
  - `for name in word {word}*`
  - `do`
  - `commands`
  - `done`

- `while test`
  - `do`
  - `commands`
  - `done`

- `until test`
  - `do`
  - `commands`
  - `done`
Looping structures, cont.

Example:
```
#!/bin/csh
foreach person (Omar Khaled Ali Faycal)
    echo Hello $person
end
```

Output:
Hello Omar
Hello Khaled
Hello Ali
Hello Faycal
Looping structures, cont.

Other example: Adding integers from 1 to 10

```bash
#!/bin/csh
set i=1
set sum=0
while ($i <= 10)
    echo Adding $i into the sum.
    set sum=`expr $sum + $i`
    set i=`expr $i + 1`
end
echo The sum is $sum.
```
Examples

1.
```bash
#!/bin/csh -f
set num_args = $#argv
echo "The number of arguments in argv is: $num_args"

set n = 0
till ( $n <= $num_args )
  echo "Argument $n is $argv[$n]"
  set n = ($n + 1)
end
```

2.
```bash
if (-f ln.exe && -f raz.exe) echo "yes"
if (-f ln.exe || -f raz.exe) echo "no"
```
Examples, cont.

3.

#!/bin/sh

# make a directory
mkdir /home/omar/mydocs

# copy all doc files
cp *.docs /home/omar/mydocs

# delete all docs files
rm *.docs
Examples, cont.

4.
#!/bin/sh
if grep `omar>` data.file > /dev/null 2>&1 then
    echo "omar is in file"
else
    echo "No, omar is not in file"
fi
/dev/null: the system dustbin
**Special shell variables**

- `#$`: the number of arguments passed to the script
- `$*`: shows in a single string all the arguments passed to the script
- `$ $$`: the current PID of the script running
- `$!`: the PID of the last process that was put in the background
- `$@`: the same as `#$`, but when used with quotes, returns each argument in quotes
- `$?`: shows the exit status of the last command. 0 is no errors, any other value is an error.
Special shell variables, cont.

echo “this is the script name” : $0
echo “this is the first parameter” : $1
echo “this is the second parameter” : $2
echo “this is the third parameter” : $3
echo “this is the fourth parameter” : $4
echo “the number of arguments passed” : $#
echo “show all arguments” : $*
echo “show me my process ID” : $$
echo “show me the arguments in quotes” : @
echo “did my script go with any errors” : $?
5. #!/bin/sh  #This script must be corrected and improved …

#is the EDITOR set?
if [-z $EDITOR]; then
    echo “Your editor environment is not set”
    echo “I will assume you want to use vi ... OK”
    echo –n “Do you wish to change it now? [y..n]:”
    read ANS
    # check for an upper or lower case ‘y’
    if [“$ANS”=“y”] || [“$ANS”=“Y”]; then
        echo “enter your editor type:”
        read EDITOR
        if [-z $EDITOR] || [“$EDITOR”=“”]; then
            echo “No, edito entered, using vi as default”
            EDITOR=vi
            export EDITOR
        fi
    fi
    # got a value use it for EDITOR
    EDITOR=$EDITOR
    export EDITOR
    echo “setting $EDITOR”
else
    #user
    echo “Using vi as the default editor”
    EDITOR=vi
    export EDITOR
fi
Examples, cont.

6. 
#!/bin/sh
BAK=".bak"
for loop in ‘ls ‘
do
    echo “copying $loop to $loop$BAK”
    cp $loop $loop$BAK
Done

7. 
#!/bin/sh
for files in ‘ls LPSO*’
do
    cat $files | tr “[a-z]” “[A-Z]” >$files.UC
done