L04 – C Shell Scripting - Part 2

1. Control Structures: if then else

Last time we worked on the basics of putting together a C Shell script. Now, it is time to add to this the control structures that actually make scripting useful.

The following example shows the three primary examples of how to test conditionally.

```
#!/bin/csh

echo "Enter a number between 1 and 10… "
@ number = $<

if ($number == 6) then
  echo "that's the lucky number!"
endif


if ($number > 5 && $number < 7) then
  echo "that's the lucky number!"
else
  echo "you lose.  try again."
endif

if ($number > 0 && $number < 5) then
  echo "a low pick."
else if ($number >= 7 && $number <= 10) then
  echo "a high pick."
else if ($number == 6) then
  echo "that's the lucky number!"
else
  echo "you didn't pick a number between 1 and 10!"
  echo "follow the instructions and try again..."
endif
```

Remember though, when testing numbers in a C Shell script, it can not handle real numbers!

2. Control Structures: goto

I shudder to actually write down the goto statement. It is, in my opinion, an abomination. It was relegated obsolete back in the 60's, yet here it is, still in existence in a handful of languages. Here are a couple of quick examples on how to use it, and then I wash my hands of it!

First, let's just look at the example given above, and put a **goto** statement in, such that if you choose a number outside of the range 1 to 10 the script will force you to re-pick a number.

```
#!/bin/csh

select:
echo "Enter a number between 1 and 10… "
@ number = $<


if ($number > 0 && $number < 5) then
  echo "a low pick."
else if ($number >= 7 && $number <= 10) then
  echo "a high pick."
else if ($number == 6) then
  echo "that's the lucky number!"
else
  echo "you didn't pick a number between 1 and 10!"
  echo "follow the instructions and try again..."
  goto select
endif
```

The following example shows how one could test for the proper usage of a C Shell script:

```
#!/bin/csh
#
# Example script requires 2 command line arguments
#  1) the name of an input file, 2) the name of an output file

if ($#argv < 2) goto usage

set ifile = $argv[1]
set ofile = $argv[2]

exit 1
usage:
echo "Usage: myprog input_file output_file"
```

My hands are clean.

3. Control Structures: loops

Once you can loop you are pretty much set. There are two main ways to loop in a C Shell: either with a **while** or a **foreach** statement. Examples of each are given below.

Example of using a **while** statement:

```
#!/bin/csh

#Example of looping through a list of files.
#
# e.g., imagine I have a bunch of SAC files that all end with the
#   suffix .R
#   i.e., I have them all rotated to the radial component.
#    Now I want to do something with those files, in this example
#   use SAC to cut them.

#make a temporary file listing all of my .R files
ls *.R >! file_list

# find out how many files I have
@ nr = `awk 'END {print NR}' file_list`

@ n = 1        # define a looping variable

# start the loop
while ($n <= $nr)

#grab nth file name from the list
set if = `awk 'NR == '$n' {print $1}' file_list`

echo "cutting file $if .."

sac << eof
r $if
cuterr fillz
cut 0 200
r
w over
q
eof

@ n = $n + 1   #increase n by one
end   # end loop

# clean up temporary files
rm file_list
```

Example of using a **foreach** statement:

```
#!/bin/csh

set phase_list = (ScP PcP P)
set depths = (100.0 200.0 300.0 400.0 500.0 600.0)

# loop through all seismic phases and depths set above
foreach phase ($phase_list)
  foreach depth ($depths)
    echo $phase $depth
  end
end
```

4. Control Structures: Switch Case

This is a really nice structure that is similar to an **if then** type of structure. Suppose I wanted to do some action based on what kind of seismic arrival I was looking at. So, if I was interested in a PKP arrival I could write some code that did tests like:

```
if ($some_string == 'PKP') then
  do something…
else if ($some_string == 'SKS') then
  do something else
else
  do another something else
endif
```

OK, a more elegant way to do this is to use the Switch Case structure:

```
#!/bin/csh

set input_phase = PKP

switch ($input_phase)

  case PKP:
    echo "PKP arrival"
    breaksw

  case SKS:
    echo "SKS arrival"
    breaksw

  case SPdKS:
    echo "SPdKS arrival"
    breaksw

endsw
```

5. Control Structures: if then else revisited

Sometimes to make your scripts more robust it is useful to do some checks before you actually implement some action. For example, no sense in trying to move the file named *blah*, if the file *blah* doesn't even exist.

To see how this works, create a temporary file named: **example.txt** and a temporary directory named: **ExampleDir**.

So, let's do some tests on these temporary files (in the Linux system directories are really just files as well).

```
#!/bin/csh

set if = example.txt      # so we don't have to type out the
                          # filename a bunch of times.
set id = ExampleDir       # as above…

if (-e  $if) then
  echo "the file $if  exists!"
endif

if (-e $id) then
  echo "$id exists!"
endif

if (-f $id) then
  echo "$id is a normal file"
else
  echo "$id is NOT normal."
endif

if (-d $id) then
  echo "$id is a directory!"
endif
```

The table below shows all of the attributes one may search for relating to files:

Letter	Attribute
d	The file is a directory file.
e	The file exists.
f	The file is an ordinary file.
o	The user owns the file.
r	The user has read access to the file.
w	The user has write access to the file.
x	The user has execute access to the file.
z	The file is 0 bytes long.

6. The Dialog utility

Let's wrap up our lectures on C Shell scripting with an entertaining utility. Perhaps you want to impress your advisor and make him/her think you've already developed these mad hacking skills. Well, try asking for input using the dialog utility. I guarantee that you will impress the entire faculty in this Dept. (with the exception of me of course).

As a quick demo:

```
#!/bin/csh

dialog --title "-----  WARNING  -----" \
--infobox "This computer will explode \
unless you press a key within the next 5 seconds!" 7 50;
set exit_status = $?
```

The dialog utility uses the following syntax:

```
dialog --title {title} --backtitle {backtitle} {Box options}

where Box options can be one of the following (other options also exist
if you check out the man page)
 --yesno      {text}  {height} {width}
 --msgbox     {text}  {height} {width}
 --infobox    {text}  {height} {width}
 --inputbox   {text}  {height} {width} [{init}]
 --textbox    {file}  {height} {width}
 --menu       {text}  {height} {width} {menu} {height} {tag1} item1}...
```

Here is an example of how to create a yes/no box:

```
#!/bin/csh

set ifile = 'blah.txt'

dialog --title "-----  Yes/No Example  -----" \
  --yesno "Do you want to delete file $ifile" 7 60

set exit_status = $?   # get the dialog utilities exit status

echo " "

switch ($exit_status)

  case 0:
  #user selected 'yes'
  echo "Deleting file $ifile"
  rm $ifile
  breaksw
```

```
  case 1:
  #user selected 'no'
  echo "Saving file $ifile"
  breaksw

  case 255:
  #user hit escape key
  echo "Operation Canceled…"
  breaksw

endsw
```

As a final example of the dialog utility, let's use it to grab some text from the user. In this example we will prompt the user to type in a file name to delete:

```
#!/bin/csh

dialog -- title "-----  Text Input Example  -----" \
   -- inputbox "Enter the name of the file you want to delete" \
   7 60 'file'  \
   --stdout > temp_menu.txt

set exit_status = $?   #get the dialog utilities exit status

#get the string that the user typed in the input box
set ifile = `cat temp_menu.txt`

echo " "

switch ($exit_status)

    case 0:
    #A file name was entered
    echo "Deleting file $ifile"
    breaksw

    case 1:
    #The cancel button was pressed
    echo "Cancel button pressed"
    breaksw

    case 255:
    #User hit the escape key
    echo "Escape key pressed"
    breaksw

endsw

rm temp_menu.txt  #get rid of temporary files
```

7. Debugging C Shell Scripts

There are two quick ways in which one can debug a C Shell script. The script can either be run from the command line as in one of the following two examples:

```
>> csh –x myscript
>> csh –v myscript
```

or, the top most line of the script can be written as follows:

```
#!/bin/csh –x
#!/bin/csh –v
```

The –x option echoes the command line after variable substitution.
The –v option echoes the command line before variable substitution.

8. Homework

1) Write a C Shell script that will loop through a list of files, and add a counter to the beginning of the filename. For example, if I have 10 files named:

```
a.txt
b.txt
c.txt
…
j.txt
```

The code should move the files to be named:

```
01_a.txt
02_b.txt
03_c.txt
…
10_j.txt
```

This kind of utility is often needed in naming files. Especially, as we will see in later lectures when automatically generating animations or movie files.

2) Write a C Shell script that will repeat a command many times. We will call this script: **forever**. For example, sometimes I want to see if a job I submitted to the supercomputer has started yet. To do so I would type **qstat –a**. Well, I'm anxious to see if it starts, so I will keep typing **qstat –a** until I get confirmation that indeed the job did start. Instead I want to type **forever qstat –a**, and what should happen is that **qstat –a** keeps getting invoked (after a couple seconds delay) until I decide to cancel it. Your script should be able to take any Unix command as input. For example, it should work as **forever ls**, or **forever ls –la**, or **forever cat inputfile**, etc.

3) In the C Shell one can not do floating point operations. That is, you can not do math with real numbers. However, it is sometimes necessary to do so. A quick work around is to do the math inside a program like the basic calculater (e.g., use: **bc -l**). Write a shell script that will allow you to do a simple calculation on floating point numbers. Take as input a coordinate position in polar coordinates (Radius, and angle theta in degrees) and output the equivalent Cartesian coordinate position.

4) Write a C Shell script using the dialog utility to create a menu box. The menu box should provide several options of actions you want to carry out on a seismogram. For example, the menu box may have options as follows:

Please choose an Action to be performed on Seismogram:

1 Flip Polarity of Seismogram
2 Low Pass Filter Seismogram
3 Make Time Picks on Seismogram
4 Discard Seismogram

The script doesn't actually have to perform any actions on a seismogram file, but is aimed at getting you to write a script using the dialog utility. Output, in the form of some kind of recognition of which option was chosen should be provided in the code.