Bourne again shell programming

- Learning the bash shell, by Cameron Newham & Bill Rosenblatt, Published by O'Reilly, ISBN 1-56592-347-2
- bourne shell provides us with a number of flow control constructs
  - `if`, `elif`, `else`, `fi`
  - `while`
  - `for`, `do`, `done`
  - `case`
    - C switch
  - `select`
    - menu selection
- `select` is not standard bourne shell so beware of using it

if elif else fi

- simplest type of flow control and has syntax:
- ```
  if condition ; then
     statements
  elif condition ; then
     statements
  else
     statements
  fi
  ```
- the *condition* is really a list of statements which returns an exit status
- in GNU/Linux an exit status of zero indicates no error
  - `if` checks the status of the last statement

if then else fi example

- guideline: *try and write your if statements to conform to the following style:*
- ```
  if command ran successfully ; then
     normal processing
  else
     error processing
  fi
  ```
  - for example
- ```
  #!/bin/sh

  HOST=mcgreg.comp.glam.ac.uk

  if ping -c 1 $HOST >& /dev/null ; then
     echo "$HOST is alive and kicking"
  else
     echo "$HOST is probably unwell"
  fi
  ```


Explanation of the script

- note that the `>& /dev/null` sends all output
  - both stderr and stdout to a pseudo device null
  - null reads any amount of input and throws it away (swallows it)
- the `#!/bin/sh` informs GNU/Linux to run your script with `/bin/sh`
- `ping -c 1 mcgreg.comp.glam.ac.uk` sends a single packet to mcgreg.comp.glam.ac.uk and waits for a reply
  ```

Expressions

- multiple statements can be combined via && and ||
  - common use:

```
if statement1 && statement2 ; then
   echo "both statement1 and \
         statement2 executed correctly"
fi
```

```
if statement1 || statement2 ; then
   echo "either statement1 ran and \
         statement2 was never executed or \
         statement1 failed and statement2 \
         executed correctly"
fi
```


Expressions

- a common use of the previous or || condition is to write the following code:

```
function die () {
   echo "die: because $1 failed"
   exit 1
}

if statement1 || die "statement1" ; then
   echo "everything is ok"
fi
```

- and make `die` a function which never returns and terminates the script

Condition tests

- the `test` ability in an expression is quite powerful
- test can either be invoked as

```
if test -x /usr/bin/gcc ; then
   echo gcc exists on this system
fi
```

  - or via:

```
if [ -x /usr/bin/gcc ] ; then
   echo gcc exists on this system
fi
```

- they are identical, people tend to prefer the later as it looks more like an expression

More condition tests

- for a definitive list of condition tests
  - type `man test` when logged into GNU/Linux

test synopsis	
operator	true if ..
str1 = str2	str1 matches str2
str1 != str2	str1 does not match str2
str1 < str2	str1 is less than str2
str1 > str2	str1 is greater than str2
-n str1	str1 is not null
-z str1	str1 is null (length 0)
-f str1	file str1 exists
-x str1	file str1 exists and is executable
str1 -ot str2	file str1 is older than file str2
str1 -nt str1	file str1 is newer than file str2
-d str1	a directory called str1 exists

while

- has the syntax:
- ```
  while condition ; do
     statements
  done
  ```
  - we might write a simple administration script as:
- ```
  while !ping -c 1 mcgreg.comp.glam.ac.uk ; do
      echo "hmm mcgreg is not responding"
      sleep 30  # wait for 30 seconds
  done
  echo "mcgreg is alive and well"
  ```


Comment character

- note the comment character is #
  - everything to the right of the comment character is ignored

Autoftp

- we will build up a simple application using bash and expect called autoftp
  - autoftp [-p] [-v] *autoftprc*
    - `-p` probes the ftp sites and checks urls in *autoftprc* are valid
    - `-v` makes autoftp issue verbose trace logs
    - *autoftprc* file is plain text ftp urls
- autoftp will automatically download a list of files from a list of ftp urls held inside *autoftprc*

Typical autoftprc file

- might contain:
- ```
  # GNU Troff
  j228-gm.comp.glam.ac.uk/pub/groff/groff.tar.gz # this is
  j228-gm.comp.glam.ac.uk/pub/groff/README       # another
  ```
  ```

Tutorial Exercise

- write a shell script program to emit a simple message onto the screen:

```
#! /bin/sh

cat << EOF
This is my simple message which will
be echoed to the screen
EOF
```

- hint read through the notes titled: Setting the scene ⟨1.html⟩

Tutorial Exercise

- write a small shell script to test whether a user has used OpenOffice
  - you can tell whether a user has invoked StarOffice as there will be a directory `OpenOffice.org1.1` in their home directory (`$HOME`)

Laboratory Exercise

- suppose your shell script is run every time a user logs into GNU/Linux
  - extend your program so that it issues the following friendly user message if the user has *never* used OpenOffice

> You might be interested in OpenOffice which is a Microsoft WORD compatible.

Laboratory

- using xemacs or an editor you are familiar with, create a file called autoftprc and type our example autoftprc contents
- now create another file called, autoftp.sh. Type in the following

```
#!/bin/sh

cut -d '#' -f1 $1
```

- make it executable and then type: `./autoftp.sh autoftprc`
  - write down the output, what has your program done?
  - what is $1 ?
  - hint try executing your program using debugging:
  - `sh -x autoftp.sh autoftprc`

Extending autoftp.sh

- now modify your program to look like this:

```
#!/bin/sh

TMP=/tmp/auto.$$

cut -d '#' -f1 $1 > $TMP
n=`wc -l $TMP | cut -b 0-7`
i=1
while [ $i -le $n ]
do
   line=`head -$i $TMP | tail -1`
   if [ "$line" != "" ]
   then
       SITE=`echo $line | cut -d '/' -f 1`
       FILE=`echo $line | cut -d '/' -f 2-`
       echo "site is" $SITE "file is" $FILE
   fi
   i=`expr $i + 1`
done

/bin/rm -f $TMP
```


Extending autoftp.sh

- run it using: `./autoftp.sh autoftprc`
  - it should produce the following output

```
site is floppsie.. file is pub/groff/groff.tar.gz
site is floppsie.. file is pub/m2/m2f-3.8.tar.gz
```

  - run it again, this time with
    `sh -x ./autoftp.sh autoftprc`
  - trace though the output and the script code
  - examine both carefully and slowly