Software Tools

C, Unix (Linux), and tools

The If statement

- The various forms are summarized as follows:
  - if list; then list; [elif list; then list;]...[else list;] fi
- Whereever list appears it means a series of commands separated by semicolon, newline, ampersand, pipe, etc.
- The exit status is the status of the last command executed or zero if no condition tested true.

Conditions

- The condition in the if or elif clauses are true if they return zero status
- Many commands like diff or cmp return a status that is expected.
- There is a host of other conditions available the most important of which is the command [
- Since it is a command it has to be followed by space

The [command

- The [command is terminated by]
- The simplest forms (unary) are
  - [-f fname], true if fname is a regular file
  - [-d dname], true if dname is a directory
  - [-x fname], true if fname is executable
  - [-s fname], true is file nfmae exists and is not empty
  - [-e fname], true if file fname exists
  - [-v varname], true if variable varname is set
  - [-z string], true if string is of zero length

The [command

- The binary operators are:
  - [str1 == str2], true if the strings are equal
  - [str1 != str2], true if not equal
  - [str1 < str2], true if str1 is less than str2 lexicographically.
  - [num1 OP num2], where OP is one of -eq, -ne, -lt, -le, -gt, or -ge, is true if the corresponding comparison is true.
  - [fname1 -nt fname2], true if fname1 is newer than fname2
  - [fname1 -ot fname2], true if fname1 is older than fname2
- There are many more, rather esoteric ones.

The [command

- There are several logic operators, too
  - [! expr], true if [expr] not true
  - [(expr)], used to override precedence
  - [expr1 -o expr2], the OR operator
  - [expr1 -a expr2], the AND operator
- One can use the && or || operators like:
  - [expr1 -o expr2] is equivalent to
  - [expr1] || [expr2]
- Just to confuse you more the [command is almost identical to the test command

The for command

- It comes in two forms
  - for var [[in [word ...]] ;] do list ; done
  - for ((expr1 ; expr2; expr3)) ; do list ; done
- The first is by far the most common
  - Variable var takes in each iteration the value of one the the words (or wildcard expansion)
  - In each iteration list is executed
  - The return status is the status of the last command executed
  - If the in clause is ommitted the positional (command line) arguments are used
  - If there is no word, then list is not executed.

The for command

- The second form is very similar to the C for statement.
- The exprs are very C like
  - for ((i=1 ; i<10 ; i++)) ; do echo $i ; done

More on Quoting

- We can quote any special character with backslash.
  - This means that, for example, a dollar is a dollar not the variable expansion operator.
- The only exception is newline. Both the backslash and the newline disappear. Used to break long lines without really breaking them.

More on Quoting

- We can quote all special characters in a string with single quotes
- A backslash is a backslash, is a backslash.
- Even the single quote itself cannot be quoted
- Used to give arguments like regular expressions that have many special characters

More on Quoting

- Double quotes allow for the dollar, back quote, and history substitution “!!” to work.
- Tha backslash retains it special status if followed by a dollar, a back quote, a double quote, a backslash, an exclamation mark, or a newline.
- Main use is to remove special status of space as a separator.