Variables

Identifiers for shell variables are similar to other languages: an alphabetic character followed by alphabetic, numeric or underscore characters.

Shell variables are all strings, although they can be used as numbers.

Builtin Variables

The shell maintains a number of variables which are built into the shell, and the values are determined by the shell.

? The value returned by the last executed command in decimal.

$ The process number of this shell.

! The process number of the last background command invoked.

- Options supplied to the shell on invocation or by `set`

0 Name of the current shell script, or the name of the shell.

The number of command line arguments in decimal.

[1-9] The command line arguments to the current shell script.

* All the command line arguments.

@ Similar to, but subtly different from '*'.

If statement

The `if` statement is similar to other programming languages, except instead of a logical expression, it takes a command and evaluates its exit status.

```
if grep -s root /etc/passwd
then
    echo ok
else
    echo we got trouble
fi
```

Or using the `test` command:

```
if [ -w / ]
then
    echo bad news
fi
```

You can also chain `else-if`'s via the `elif` construct

```
if [ $x -gt 0 ]
then
    echo positive
elif [ $x -lt 0 ]
then
    echo negative
else
    echo zero
fi
```

Gotcha
Note that the shell concept of true or false is reverse of C and other languages: 0 is true, anything else is false.

Command grouping

There are two ways to group commands:

```
{ commands ; }
```

or

```
( commands )
```

The former merely executes the commands in sequence, the latter executes the commands in a subshell. A simple example is:

```
(cd monty; rm chapman)
```

In this case the `cd` will not affect the current shell.

A more useful example is:

```
(cd /src/gnu/groff; tar cvf - ) | tar xvf -
```

- In both cases the commands should be separated by semi-colons.