A shell is a command line interpreter which can constitute the user interface for terminal windows. It can also be used as a mechanism to run scripts, even in non-interactive sessions without a terminal window, as if the commands were being typed in.

For example, typing:

```
$ find . -name "*.c" -ls
```

at the command line accomplishes the same thing as running the following script:

```
#!/bin/bash
find . -name "*.c" -ls
```

The **#!/bin/bash** at the beginning of the script should be familiar to anyone who has developed any kind of script in UNIX environments. Following the magic **#!** characters goes the name of whatever scripting language interpreter is tasked with executing the following lines. Choices include **/usr/bin/perl**, **/bin/bash**, **/bin/csh**, **/usr/bin/python** and **/bin/sh**.

Linux provides a wide choice of shells; exactly what is available to use is listed in **/etc/shells**; e.g. on one system we get:

```
$ cat /etc/shells
/bin/sh
/bin/bash
/sbin/nologin
/bin/tcsh
/bin/csh
/bin/ksh
/bin/zsh
```

Most Linux users use the default bash shell, but those with long UNIX backgrounds with other shells may want to override the default. It is worth reviewing the main choices in the historical order of introduction.

✓ Complete Go to next item