Next Up Previous
: For Loops : Shell Programming : Shell Variables

Input Output Redirection

We saw in chapter two that the standard input to and standard output from commands can be redirected with the redirection symbols <, > and ». We saw also that the output of one command can be used as input to another if the commands are connected with a pipe (denoted by the pipe symbol |).

In addition to standard input and output, commands also have a standard error stream to which they can write. By default the standard error is connected to the user's terminal. It is not redirected by the symbols > and >>. The standard error was invented so that error messages would always appear on the terminal.

```
$ cat xxx yyy > zzz
cat: xxx: No such file or directory
cat: yyy: No such file or directory
```

If the error messages were written to standard output, they would be sent to the file `zzz` and the user would be blissfully unaware that there was a problem.

The three default streams are numbered by the integers `0`, `1` and `2` corresponding to standard input, standard output and standard error. These integers are sometimes used to explicitly redirect the output streams. For example, the command

```
$ cat xxx yyy > zzz 2> errors
```

redirects the standard output of the `cat` command into `zzz` and simulaneously redirects the standard error into the file `errors`.

The following table shows the ways in which the shell's input and output streams can be manipulated.

`> file`	direct standard output to `file`
`>> file`	append standard output to `file`
`< file`	take standard input from `file`
`cmd1 | cmd2`	use standard output from `cmd1` as standard input to `cmd2`
`^`	obsolete synonym for `|`
`2> file`	direct standard error to *file*
`2>> file`	append standard error to *file*
`2>&1`	print standard error messages to standard output
`1>&2`	print standard output messages to standard error

`<< string`	here document: take standard input until next `string` at the beginning of a line; substitute for `$`, `` `...` `` and `\`
`<< \string`	here document with no substitution
`<< 'string'`	here document with no substitution

There are two idioms in the table which we have not discussed but which are relatively important.

The notation `2>&1` tells the shell to write standard error output to the standard output stream. It is also possible to redirect standard output to the standard error stream. This is most commonly used as follows to produce error messages on the standard error stream.

```
echo ··· 1>&2
```

The shell provides a mechanism for including the standard input to a command in the same file as the command rather than in a separate file. This means that a shell script can be completely self-contained. Our phone number program could be written using this mechanism as follows.

```
grep -i "$*" << END
University      373-7599
Statistics      x87510
Fax             373-7018 x87018
Ronald Fisher   x81234
Jerzy Neyman    x81235
Karl Pearson    x81236
END
```

The `<<` signals the start of the construction; the word which follows (`END` in our example) is given on a line by itself to signal the end of the input. The standard jargon for input specified in this way is a *here document*. The shell substitutes for `$`, `` `...` ``, and `\` in here documents, unless some part of the delimiting word after `<<` is quoted with quotes or a backslash. In the `phone` script above we could turn off metacharacter substitution by changing the first line of the script to

```
grep -i "$*" << 'END'
```

Next Up Previous

: For Loops : Shell Programming : Shell Variables

Ross Ihaka `'``17ž11``26`