Next: Executing Commands, Previous: Shell Expansions, Up: Basic Shell Features [Contents][Index]

3.6 Redirections

Before a command is executed, its input and output may be *redirected* using a special notation interpreted by the shell. Redirection allows commands' file handles to be duplicated, opened, closed, made to refer to different files, and can change the files the command reads from and writes to. Redirection may also be used to modify file handles in the current shell execution environment. The following redirection operators may precede or appear anywhere within a simple command or may follow a command. Redirections are processed in the order they appear, from left to right.

Each redirection that may be preceded by a file descriptor number may instead be preceded by a word of the form {*varname*}. In this case, for each redirection operator except >&- and <&-, the shell will allocate a file descriptor greater than 10 and assign it to {*varname*}. If >&- or <&- is preceded by {*varname*}, the value of *varname* defines the file descriptor to close. If {*varname*} is supplied, the redirection persists beyond the scope of the command, allowing the shell programmer to manage the file descriptor himself.

In the following descriptions, if the file descriptor number is omitted, and the first character of the redirection operator is '<', the redirection refers to the standard input (file descriptor 0). If the first character of the redirection operator is '>', the redirection refers to the standard output (file descriptor 1).

The word following the redirection operator in the following descriptions, unless otherwise noted, is subjected to brace expansion, tilde expansion, parameter expansion, command substitution, arithmetic expansion, quote removal, filename expansion, and word splitting. If it expands to more than one word, Bash reports an error.

Note that the order of redirections is significant. For example, the command

```
ls > dirlist 2>&1
```

directs both standard output (file descriptor 1) and standard error (file descriptor 2) to the file *dirlist*, while the command

```
ls 2>&1 > dirlist
```

directs only the standard output to file *dirlist*, because the standard error was made a copy of the standard output before the standard output was redirected to *dirlist*.

Bash handles several filenames specially when they are used in redirections, as described in the following table. If the operating system on which Bash is running provides these special files, bash will use them; otherwise it will emulate them internally with the behavior described below.

/dev/fd/*fd*

If *fd* is a valid integer, file descriptor *fd* is duplicated.

/dev/stdin

File descriptor 0 is duplicated.

/dev/stdout

File descriptor 1 is duplicated.

/dev/stderr

File descriptor 2 is duplicated.

/dev/tcp/*host*/*port*

If *host* is a valid hostname or Internet address, and *port* is an integer port number or service name, Bash attempts to open the corresponding TCP socket.

/dev/udp/*host*/*port*

If *host* is a valid hostname or Internet address, and *port* is an integer port number or service name, Bash attempts to open the corresponding UDP socket.

A failure to open or create a file causes the redirection to fail.

Redirections using file descriptors greater than 9 should be used with care, as they may conflict with file descriptors the shell uses internally.

3.6.1 Redirecting Input

Redirection of input causes the file whose name results from the expansion of *word* to be opened for reading on file descriptor `n`, or the standard input (file descriptor 0) if `n` is not specified.

The general format for redirecting input is:

```
[n]<word
```

3.6.2 Redirecting Output

Redirection of output causes the file whose name results from the expansion of *word* to be opened for writing on file descriptor *n*, or the standard output (file descriptor 1) if *n* is not specified. If the file does not exist it is created; if it does exist it is truncated to zero size.

The general format for redirecting output is:

```
[n]>[|]word
```

If the redirection operator is '`>`', and the `noclobber` option to the `set` builtin has been enabled, the redirection will fail if the file whose name results from the expansion of *word* exists and is a regular file. If the redirection operator is '`>|`', or the redirection operator is '`>`' and the `noclobber` option is not enabled, the redirection is attempted even if the file named by *word* exists.

3.6.3 Appending Redirected Output

Redirection of output in this fashion causes the file whose name results from the expansion of *word* to be opened for appending on file descriptor *n*, or the standard output (file descriptor 1) if *n* is not specified. If the file does not exist it is created.

The general format for appending output is:

```
[n]>>word
```

3.6.4 Redirecting Standard Output and Standard Error

This construct allows both the standard output (file descriptor 1) and the standard error output (file descriptor 2) to be redirected to the file whose name is the expansion of *word*.

There are two formats for redirecting standard output and standard error:

```
&>word
```

and

```
>&word
```

Of the two forms, the first is preferred. This is semantically equivalent to

```
>word 2>&1
```

When using the second form, *word* may not expand to a number or '`-`'. If it does, other redirection operators apply (see Duplicating File Descriptors below) for compatibility reasons.

3.6.5 Appending Standard Output and Standard Error

This construct allows both the standard output (file descriptor 1) and the standard error output (file descriptor 2) to be appended to the file whose name is the expansion of *word*.

The format for appending standard output and standard error is:

```
&>>word
```

This is semantically equivalent to

```
>>word 2>&1
```

(see Duplicating File Descriptors below).

3.6.6 Here Documents

This type of redirection instructs the shell to read input from the current source until a line containing only *word* (with no trailing blanks) is seen. All of the lines read up to that point are then used as the standard input (or file descriptor *n* if *n* is specified) for a command.

The format of here-documents is:

```
[n]<<[-]word
        here-document
delimiter
```

No parameter and variable expansion, command substitution, arithmetic expansion, or filename expansion is performed on *word*. If any part of *word* is quoted, the *delimiter* is the result of quote removal on *word*, and the lines in the here-document are not expanded. If *word* is unquoted, all lines of the here-document are subjected to parameter expansion, command substitution, and arithmetic expansion, the character sequence `\newline` is ignored, and '`\`' must be used to quote the characters '`\`', '`$`', and '`` ` ``'.

If the redirection operator is '`<<-`', then all leading tab characters are stripped from input lines and the line containing *delimiter*. This allows here-documents within shell scripts to be indented in a natural fashion.

3.6.7 Here Strings

A variant of here documents, the format is:

```
[n]<<< word
```

The *word* undergoes tilde expansion, parameter and variable expansion, command substitution, arithmetic expansion, and quote removal. Pathname expansion and word splitting are not performed. The result is supplied as a single string, with a newline appended, to the command on its standard input (or file descriptor *n* if *n* is specified).

3.6.8 Duplicating File Descriptors

The redirection operator

```
[n]<&word
```

is used to duplicate input file descriptors. If *word* expands to one or more digits, the file descriptor denoted by *n* is made to be a copy of that file descriptor. If the digits in *word* do not specify a file descriptor open for input, a redirection error occurs. If *word* evaluates to '-', file descriptor *n* is closed. If *n* is not specified, the standard input (file descriptor 0) is used.

The operator

```
[n]>&word
```

is used similarly to duplicate output file descriptors. If *n* is not specified, the standard output (file descriptor 1) is used. If the digits in *word* do not specify a file descriptor open for output, a redirection error occurs. If *word* evaluates to '-', file descriptor *n* is closed. As a special case, if *n* is omitted, and *word* does not expand to one or more digits or '-', the standard output and standard error are redirected as described previously.

3.6.9 Moving File Descriptors

The redirection operator

```
[n]<&digit-
```

moves the file descriptor *digit* to file descriptor *n*, or the standard input (file descriptor 0) if *n* is not specified. *digit* is closed after being duplicated to *n*.

Similarly, the redirection operator

```
[n]>&digit-
```

moves the file descriptor *digit* to file descriptor *n*, or the standard output (file descriptor 1) if *n* is not specified.

3.6.10 Opening File Descriptors for Reading and Writing

The redirection operator

```
[n]<>word
```

causes the file whose name is the expansion of *word* to be opened for both reading and writing on file descriptor *n*, or on file descriptor 0 if *n* is not specified. If the file does not exist, it is created.

Next: Executing Commands, Previous: Shell Expansions, Up: Basic Shell Features [Contents][Index]