

What Are the Special Variables?

In this section we simply list of the special variables you can use in your programs. The list is identical to the list in the file `PERLVAR.htm`, which should come with your Perl distribution.

- `$_` -- The default parameter for a lot of functions.
- `$.` -- Holds the current record or line number of the file handle that was last read. It is read-only and will be reset to 0 when the file handle is closed.
- `$/` -- Holds the input record separator. The record separator is usually the newline character. However, if `$/` is set to an empty string, two or more newlines in the input file will be treated as one.
- `$,` -- The output separator for the `print()` function. Normally, this variable is an empty string. However, setting `$,` to a newline might be useful if you need to print each element in the parameter list on a separate line.
- `$\` -- Added as an invisible last element to the parameters passed to the `print()` function. Normally, an empty string, but if you want to add a newline or some other suffix to everything that is printed, you can assign the suffix to `$`.
- `$#` -- The default format for printed numbers. Normally, it's set to `%.20g`, but you can use the format specifiers covered in the section "Example: Printing Revisited" in Chapter 9 to specify your own default format.
- `$%` -- Holds the current page number for the default file handle. If you use `select()` to change the default file handle, `$%` will change to reflect the page number of the newly selected file handle.
- `$=` -- Holds the current page length for the default file handle. Changing the default file handle will change `$=` to reflect the page length of the new file handle.
- `$-` -- Holds the number of lines left to print for the default file handle. Changing the default file handle will change `$-` to reflect the number of lines left to print for the new file handle.
- `$~` -- Holds the name of the default line format for the default file handle. Normally, it is equal to the file handle's name.
- `$^` -- Holds the name of the default heading format for the default file handle. Normally, it is equal to the file handle's name with `_TOP` appended to it.

`$|` -- If nonzero, will flush the output buffer after every `write()` or `print()` function. Normally, it is set to 0.

`$$` -- This UNIX-based variable holds the process number of the process running the Perl interpreter.

`$?` -- Holds the status of the last pipe close, back-quote string, or `system()` function.

`$&` -- Holds the string that was matched by the last successful pattern match.

`$`` -- Holds the string that preceded whatever was matched by the last successful pattern match.

`$'` -- Holds the string that followed whatever was matched by the last successful pattern match.

`$+` -- Holds the string matched by the last bracket in the last successful pattern match. For example, the statement `/Fieldname: (.*)|Fldname: (.*)/ && ($fName = $+);` will find the name of a field even if you don't know which of the two possible spellings will be used.

`$*` -- Changes the interpretation of the `^` and `$` pattern anchors. Setting `$*` to 1 is the same as using the `/m` option with the regular expression matching and substitution operators. Normally, `$*` is equal to 0.

`$0` -- Holds the name of the file containing the Perl script being executed.

`$<number>` -- This group of variables (`$1`, `$2`, `$3`, and so on) holds the regular expression pattern memory. Each set of parentheses in a pattern stores the string that match the components surrounded by the parentheses into one of the `$<number>` variables.

`$[` -- Holds the base array index. Normally, it's set to 0. Most Perl authors recommend against changing it without a very good reason.

`$]` -- Holds a string that identifies which version of Perl you are using. When used in a numeric context, it will be equal to the version number plus the patch level divided by 1000.

`$"` -- This is the separator used between list elements when an array variable is interpolated into a double-quoted string. Normally, its value is a space character.

`$;` -- Holds the subscript separator for multidimensional array emulation. Its use is beyond the scope of this book.

`$!` -- When used in a numeric context, holds the current value of `errno`. If used in a string context, will hold the error string associated with `errno`.

`$@`

- Holds the syntax error message, if any, from the last `eval()` function call.
- \$<**
 - This UNIX-based variable holds the read `uid` of the current process.
- \$>**
 - This UNIX-based variable holds the effective `uid` of the current process.
- \$)**
 - This UNIX-based variable holds the read `gid` of the current process. If the process belongs to multiple groups, then `$)` will hold a string consisting of the group names separated by spaces.
- \$:**
 - Holds a string that consists of the characters that can be used to end a word when word-wrapping is performed by the `^` report formatting character. Normally, the string consists of the space, newline, and dash characters.
- \$^D**
 - Holds the current value of the debugging flags. For more information.
- \$^F**
 - Holds the value of the maximum system file description. Normally, it's set to 2. The use of this variable is beyond the scope of this book.
- \$^I**
 - Holds the file extension used to create a backup file for the in-place editing specified by the `-i` command line option. For example, it could be equal to `".bak."`
- \$^L**
 - Holds the string used to eject a page for report printing.
- \$^P**
 - This variable is an internal flag that the debugger clears so it will not debug itself.
- \$^T**
 - Holds the time, in seconds, at which the script begins running.
- \$^W**
 - Holds the current value of the `-w` command line option.
- \$^X**
 - Holds the full pathname of the Perl interpreter being used to run the current script.
- \$ARGV**
 - Holds the name of the current file being read when using the diamond operator (`<>`).
- @ARGV**
 - This array variable holds a list of the command line arguments. You can use `$#ARGV` to determine the number of arguments minus one.
- @F**
 - This array variable holds the list returned from autosplit mode. Autosplit mode is associated with the `-a` command line option.
- @Inc**
 - This array variable holds a list of directories where Perl can look for scripts to execute. The list is mainly used by the `require` statement.
- %Inc**
 - This hash variable has entries for each filename included by `do` or `require` statements. The key of the hash entries are the filenames, and the values are

the paths where the files were found.

%ENV

-- This hash variable contains entries for your current environment variables. Changing or adding an entry affects only the current process or a child process, never the parent process. See the section "Example: Using the %ENV Variable" later in this chapter.

%SIG

-- This hash variable contains entries for signal handlers. For more information about signal handlers

_

-- This file handle (the underscore) can be used when testing files. If used, the information about the last file tested will be used to evaluate the new test.

DATA

-- This file handle refers to any data following `__END__`.

STDERR

-- This file handle is used to send output to the standard error file. Normally, this is connected to the display, but it can be redirected if needed.

STDIN

-- This file handle is used to read input from the standard input file. Normally, this is connected to the keyboard, but it can be changed.

STDOUT

-- This file handle is used to send output to the standard output file. Normally, this is the display, but it can be changed.