E

Command Line Reference

The following is a list of the available switches which can be appended to the calling of perl from the command line. The exact syntax for calling perl from the command line is as follows:

```
C:\Perl\bin\Perl.exe (switches) (--) (programfile) (arguments)
```

Switch	Function
`-0(octal)`	This sets the record separator (`$/`) by specifying the character's number in the ASCII table in octal. For example, if we wanted to set our separator to the character `'e'` we would say `perl -0101`. The default is the null character, and `$/` is set to this if no argument is given. See Appendix F for a complete ASCII table.
`-a`	`-a` can be used in conjunction with `-n` or `-p`. It enables autosplit, and uses whitespace as the default delimiter. Using `-p` will print out the results, which are always stored in the array `@F`.
`-C`	Enables native wide character system interfaces
`-c`	This is a syntactic test only. It stops Perl executing, but reports on any compilation errors that a program has before it exits. Any other switches that have a runtime effect on your program will be ignored will `-c` is enabled.
`-d filename`	This switch invokes the Perl debugger. The Perl debugger will only run once you have gotten your program to compile. Enabling `-d` allows you to prompt debugging commands such as install breakpoints, and many others.

Table continued on following page

Switch	Function

`-D(number)`
 	
`-D(list)`
	`-D` will set debugging flags, but only if you have debugging compiled into your program. The following table shows you the arguments that you may use for `-D`, and the resulting meaning of the switch.
	
	Argument (number)
	
	Argument (character)
	
	Operation
	
	`1`
	
	`p`
	
	Tokenizing and parsing
	
	`2`
	
	`s`
	
	Stack snapshots
	
	`4`
	
	`l`
	
	Label stack Processing
	
	`8`
	
	`t`
	
	Trace execution
	
	`16`
	
	`o`
	
	Object method lookup
	
	`32`
	
	`c`
	
	String/numeric conversions

Switch	Function

`-D(list)` (cont.)	`64`
	`p`
	Print preprocessor command for `-p`
	
	`128`
	`m`
	Memory allocation
	
	`256`
	`f`
	Format processing
	
	`512`
	`r`
	Regular expression processing
	
	`1024`
	`x`
	Syntax tree dump
	
	`2048`
	`u`
	Tainting checks
	
	`4096`
	`l`
	Memory leaks
	
	`8192`
	`h`
	Hash dump

Table continued on following page

Switch	Function
-D(list) (cont.)	16384 X Scratchpad allocations 32768 D Cleaning up
-e	This allows you to write one line of script – by instructing Perl to execute text following the switch on the command line – without loading and running a program file. Multiple calls may be made to -e in order to build up scripts of more than one line.
-F/pattern/	Causes -a to split using the pattern specified between the delimiters. The delimiters may be / /, " ", or ' '.
-h	Prints out a list of all the command line switches.
-i(extension)	Modifies the <> operator. Makes a backup file if an argument is given. The argument is treated as the extension the saved file is to be given.
-I (directory)	Causes a directory to be added to the search path when looking for files to include. This path will be searched before the default paths, one of which is the current directory, the other is generally /usr/local/lib/Perl on Unix and C:\perl\bin on Windows.
-l(octal)	-l adds line endings, and defines the line terminator by specifying the character's number in the ASCII table in octal. If it is used with -n or -p, it will chomp the line terminator. If the argument is omitted, then $\ is given the current value of $/. The default value of the special variable $/ is newline.
-(mM)(-)module	Causes the import of the given module for use by your script, before executing the program.
-n	Causes Perl to assume a while (<>) {My Script} loop around your script. Basically it will iterate over the filename arguments. It does no printing of lines.
-p	This is the same as -n, except it will print lines.
-P	-P causes your program to be run through the C preprocessor before it is compiled. Bear in mind that the preprocessor directives begin with #, the same as comments, so rather use ;# to comment your script when you use the -P switch.

Switch	Function
`-s`	This defines variables with the same name as the switches that follow on the command line. The other switches are also removed from `@ARGV`. The newly defined variables are set to 1 by default. Some parsing of the other switches is also enabled.
`-S`	Causes perl to look for a given program file using the `PATH` environment variable. In other words, it acts much like `#!`
`-T`	Stops data entering a program from performing unsafe operations. It's a good idea to use this when there is a lot of information exchange occuring, like in CGI programming.
`-u`	This will perform a core dump after compiling the program.
`-U`	This forces Perl to allow unsafe operations.
`-v`	Prints the version of Perl that is currently being used (includes VERY IMPORTANT perl info).
`-V(:variable)`	Prints out a summary of the main configuration values used by Perl during compiling. It will also print out the value of the `@INC` array.
`-w`	Invokes the rasiing of many useful warnings based on the (poor or bad) syntax of the program being run. This switch has been deprecated in perl 5.6, in favor of the `use warnings` pragma.
`-W`	Enables **all** warnings.
`-x(directory)`	Tells Perl to get rid of extraneous text that precedes the shebang line. All switches on the shebang line will still be enabled.
`-X`	This will disable **all** warnings. We already know that we always use `use warnings` when writing our programs. So you won't need this.