Command line options ¶

The list of command line options provided by the PHP binary can be queried at any time by running PHP with the **-h** switch:

```
Usage: php [options] [-f] <file> [--] [args...]
   php [options] -r <code> [--] [args...]
   php [options] [-B <begin_code>] -R <code> [-E <end_code>] [--] [args...]
   php [options] [-B <begin_code>] -F <file> [-E <end_code>] [--] [args...]
   php [options] -- [args...]
   php [options] -a

  -a               Run interactively
  -c <path>|<file> Look for php.ini file in this directory
  -n               No php.ini file will be used
  -d foo[=bar]     Define INI entry foo with value 'bar'
  -e               Generate extended information for debugger/profiler
  -f <file>        Parse and execute <file>.
  -h               This help
  -i               PHP information
  -l               Syntax check only (lint)
  -m               Show compiled in modules
  -r <code>        Run PHP <code> without using script tags <?..?>
  -B <begin_code>  Run PHP <begin_code> before processing input lines
  -R <code>        Run PHP <code> for every input line
  -F <file>        Parse and execute <file> for every input line
  -E <end_code>    Run PHP <end_code> after processing all input lines
  -H               Hide any passed arguments from external tools.
  -S <addr>:<port> Run with built-in web server.
  -t <docroot>     Specify document root <docroot> for built-in web server.
  -s               Output HTML syntax highlighted source.
  -v               Version number
  -w               Output source with stripped comments and whitespace.
  -z <file>        Load Zend extension <file>.

  args...          Arguments passed to script. Use -- args when first argument
                   starts with - or script is read from stdin

  --ini            Show configuration file names

  --rf <name>      Show information about function <name>.
  --rc <name>      Show information about class <name>.
  --re <name>      Show information about extension <name>.
  --rz <name>      Show information about Zend extension <name>.
  --ri <name>      Show configuration for extension <name>.
```

Command line options

Option	Long Option	Description
-a	--interactive	Run PHP interactively. For more information, see the Interactive shell section.
-b	--bindpath	Bind Path for external FASTCGI Server mode (CGI only).
-C	--no-chdir	Do not chdir to the script's directory (CGI only).
-q	--no-header	Quiet-mode. Suppress HTTP header output (CGI only).
-T	--timing	Measure execution time of script repeated *count* times (CGI only).
-c	--php-ini	Specifies either a directory in which to look for *php.ini*, or a custom *INI* file (which does not need to be named *php.ini*), e.g.: `$ php -c /custom/directory/ my_script.php` `$ php -c /custom/directory/custom-file.ini my_script.php` If this option is not specified, *php.ini* is searched for in the default locations.

-n --no-php-ini

Ignore *php.ini* completely.

-d --define

Set a custom value for any of the configuration directives allowed in *php.ini*. The syntax is:

```
 -d configuration_directive[=value]
```

```
# Omitting the value part will set the given configuration directive to "1"
$ php -d max_execution_time
        -r '$foo = ini_get("max_execution_time"); var_dump($foo);'
string(1) "1"

# Passing an empty value part will set the configuration directive to ""
php -d max_execution_time=
        -r '$foo = ini_get("max_execution_time"); var_dump($foo);'
string(0) ""

# The configuration directive will be set to anything passed after the '=' character
$  php -d max_execution_time=20
        -r '$foo = ini_get("max_execution_time"); var_dump($foo);'
string(2) "20"
$  php
        -d max_execution_time=doesntmakesense
        -r '$foo = ini_get("max_execution_time"); var_dump($foo);'
string(15) "doesntmakesense"
```

-e --profile-info

Activate the extended information mode, to be used by a debugger/profiler.

-f --file

Parse and execute the specified file. The **-f** is optional and may be omitted - providing just the filename to execute is sufficient.

> **Nota**:
>
> To pass arguments to a script, the first argument must be --, otherwise PHP will interpret them as PHP options.

-h and -? --help and --usage

Output a list of command line options with one line descriptions of what they do.

-i --info

Calls phpinfo(), and prints out the results. If PHP is not working correctly, it is advisable to use the command **php -i** and see whether any error messages are printed out before or in place of the information tables. Beware that when using the CGI mode the output is in HTML and therefore very large.

-l --syntax-check

Provides a convenient way to perform only a syntax check on the given PHP code. On success, the text *No syntax errors detected in <filename>* is written to standard output and the shell return code is *0*. On failure, the text *Errors parsing <filename>* in addition to the internal parser error message is written to standard output and the shell return code is set to *-1*.

This option won't find fatal errors (like undefined functions). Use the **-f** to test for fatal errors too.

> **Nota**:
>
> This option does not work together with the **-r** option.

-m --modules

Exemplo #1 Printing built in (and loaded) PHP and Zend modules

```
$ php -m
[PHP Modules]
xml
tokenizer
standard
session
posix
pcre
overload
mysql
mbstring
ctype

[Zend Modules]
```

-r --run

Allows execution of PHP included directly on the command line. The PHP start and end tags (*<?php* and *?>*) are *not needed* and will cause a parse error if present.

> **Nota**:
>
> Care must be taken when using this form of PHP not to collide with command line variable substitution done by the shell.
>
> **Exemplo #2 Getting a syntax error when using double quotes**
>
> ```
> $ php -r "$foo = get_defined_constants();"
> PHP Parse error:  syntax error, unexpected '=' in Command line code on line 1
>
> Parse error: syntax error, unexpected '=' in Command line code on line 1
> ```
>
> The problem here is that sh/bash performs variable substitution even when using double quotes *"*. Since the variable *$foo* is unlikely to be defined, it expands to nothing which results in the code passed to PHP for execution actually reading:
>
> ```
> $ php -r " = get_defined_constants();"
> ```
>
> The correct way would be to use single quotes *'*. Variables in single-quoted strings are not expanded by sh/bash.
>
> **Exemplo #3 Using single quotes to prevent the shell's variable substitution**
>
> ```
> $ php -r '$foo = get_defined_constants(); var_dump($foo);'
> array(370) {
>   ["E_ERROR"]=>
>   int(1)
>   ["E_WARNING"]=>
>   int(2)
>   ["E_PARSE"]=>
>   int(4)
>   ["E_NOTICE"]=>
>   int(8)
>   ["E_CORE_ERROR"]=>
>   [...]
> ```
>
> If using a shell other than sh/bash, further issues might be experienced - if appropriate, a bug report should be opened at » http://bugs.php.net/. It is still easy to run into trouble when trying to use variables (shell or PHP) in commnad-line code, or using backslashes for escaping, so take great care when doing so. You have been warned!

> **Nota**:
>
> **-r** is available in the CLI SAPI, but not in the *CGI* SAPI.

> **Nota**:
>
> This option is only intended for very basic code, so some configuration directives (such as auto_prepend_file and auto_append_file) are ignored in this mode.

-B --process-begin

PHP code to execute before processing stdin. Added in PHP 5.

-R --process-code

PHP code to execute for every input line. Added in PHP 5.

There are two special variables available in this mode: *$argn* and *$argi*. *$argn* will contain the line PHP is processing at that moment, while *$argi* will contain the line number.

-F --process-file

PHP file to execute for every input line. Added in PHP 5.

-E --process-end

PHP code to execute after processing the input. Added in PHP 5.

Exemplo #4 Using the -B , -R and -E options to count the number of lines of a project.

```
$ find my_proj | php -B '$l=0;' -R '$l += count(@file($argn));' -E 'echo "Total Lines: $l\n";'
Total Lines: 37328
```

-S --server

Starts built-in web server. Available as of PHP 5.4.0.

-t --docroot

Specifies document root for built-in web server. Available as of PHP 5.4.0.

-s --syntax-highlight and --syntax-highlighting

Display colour syntax highlighted source.

This option uses the internal mechanism to parse the file and writes an HTML highlighted version of it to standard output. Note that all it does is generate a block of *<code> [...] </code>* HTML tags, no HTML headers.

> **Nota**:
>
> This option does not work together with the **-r** option.

-v --version

Exemplo #5 Using -v to get the SAPI name and the version of PHP and Zend

```
$ php -v
PHP 5.3.1 (cli) (built: Dec 11 2009 19:55:07)
Copyright (c) 1997-2009 The PHP Group
Zend Engine v2.3.0, Copyright (c) 1998-2009 Zend Technologies
```

-w --strip

Display source with comments and whitespace stripped.

> **Nota**:
>
> This option does not work together with the **-r** option.

-z --zend-extension

Load Zend extension. If only a filename is given, PHP tries to load this extension from the current default library path on your system (usually */etc/ld.so.conf* on Linux systems, for example). Passing a filename with an absolute path will not use the system's library search path. A relative filename including directory information will tell PHP to try loading the extension relative to the current directory.

--ini

Show configuration file names and scanned directories. Available as of PHP 5.2.3.

Exemplo #6 --*ini* example

```
$ php --ini
Configuration File (php.ini) Path: /usr/dev/php/5.2/lib
Loaded Configuration File:         /usr/dev/php/5.2/lib/php.ini
Scan for additional .ini files in: (none)
Additional .ini files parsed:      (none)
```

--rf --rfunction

Show information about the given function or class method (e.g. number and name of the parameters). Available as of PHP 5.1.2.

This option is only available if PHP was compiled with Reflection support.

Exemplo #7 basic --*rf* usage

```
$ php --rf var_dump
Function [ <internal> public function var_dump ] {

  - Parameters [2] {
    Parameter #0 [ <required> $var ]
    Parameter #1 [ <optional> $... ]
  }
}
```

Show information about the given class (list of constants, properties and methods). Available as of PHP 5.1.2.

This option is only available if PHP was compiled with Reflection support.

--rc --rclass

Exemplo #8 --*rc* example

```
$ php --rc Directory
Class [ <internal:standard> class Directory ] {

  - Constants [0] {
  }

  - Static properties [0] {
  }

  - Static methods [0] {
  }

  - Properties [0] {
  }

  - Methods [3] {
    Method [ <internal> public method close ] {
    }

    Method [ <internal> public method rewind ] {
    }

    Method [ <internal> public method read ] {
    }
  }
}
```

--re --rextension

Show information about the given extension (list of *php.ini* options, defined functions, constants and classes). Available as of PHP 5.1.2.

This option is only available if PHP was compiled with Reflection support.

Exemplo #9 --*re* example

```
$ php --re json
Extension [ <persistent> extension #19 json version 1.2.1 ] {

  - Functions {
    Function [ <internal> function json_encode ] {
    }
    Function [ <internal> function json_decode ] {
    }
  }
}
```

--rz --rzendextension

Show the configuration information for the given Zend extension (the same information that is returned by phpinfo()). Available as of PHP 5.4.0.

--ri --rextinfo

Show the configuration information for the given extension (the same information that is returned by phpinfo()). Available as of PHP 5.2.2. The core configuration information is available using "main" as extension name.

Exemplo #10 --*ri* example

```
$ php --ri date

date

date/time support => enabled
"Olson" Timezone Database Version => 2009.20
Timezone Database => internal
Default timezone => Europe/Oslo

Directive => Local Value => Master Value
date.timezone => Europe/Oslo => Europe/Oslo
date.default_latitude => 59.930972 => 59.930972
date.default_longitude => 10.776699 => 10.776699
date.sunset_zenith => 90.583333 => 90.583333
date.sunrise_zenith => 90.583333 => 90.583333
```

Nota:

Options *-rBRFEH*, *--ini* and *--r[fcezi]* are available only in CLI.

add a note

User Contributed Notes

There are no user contributed notes for this page.