


# Relative and absolute paths, in the file system and on the web server.

- 1. Intro
- 2. The difference between absolute and relative paths
- 3. Absolute paths
- 4. Relative paths
- 5. Document root
- 6. Web server paths
- 7. Console scripts. Single entry point
- 8. Helpful PHP commands and constants
-  Comments (9)

## Intro

Your site exists in two realms at once: the real and the virtual one.

For the site visitors it's entirely a **virtual** server, which in many ways is different from a real one. *There are no files for starter.* I know, it's hard to believe at first, but it's a fact. In the address like `http://example.com/file.html`, `file.html` is not a file. It's a part of URI, a virtual resource. There could be or could be not a real file with such a name, but it doesn't matter. Your browser cannot know that, and don't need to. All it needs to know is an address.

For the site developer, on the other hand, their site is a certain program running on a particular server, on the very real computer with HDD, files and directories. And your PHP script, while reading data files or including other scripts, is working with such real files that exist on the *physical medium*.

So this dualism is the root of many problems.

PHP users confuse these matters badly at first, doing things like being unable to locate an existing file, confusing hyperlinks with files, including local files via HTTP and such.

However, to sort these things out all you need is to grasp just two simple concepts:

- 1. The difference between **absolute** and **relative** paths.
- 2. The difference between the **root of the web server** and the **filesystem root**.

## The difference between absolute and relative paths

It's fairly simple.

- If the path is built starting from the system **root**, it is called **absolute**.
- If the path is built starting from the current location, it is called **relative** (which makes sense, as it is relative to our present position)

It's exactly the same as with the real life directions. Given the absolute address, a postal one, like "7119 W Sunset Blvd West Hollywood, CA 90046" you can find the location from anywhere. However, given the relative directions, like "keep three blocks this way and then and turn to the right" would work from the current location only, otherwise sending you astray.

So it goes for the paths in the computer world: given the absolute address, you can always get to the place, no matter from where you started. Whereas relative path is tricky, and should be used with caution, only when you positively know where you are at the moment.

## Absolute paths

So again: an absolute path is one starting from the system root

Some absolute path examples:

```
/var/www/site/forum/index.php
/img/frame.gif
C:\windows\command.com
```

Note that in Unix-like systems (and web-servers) the root is defined as a slash - `/`. And this is very important to know. It is not just a marker, but already a full qualified address, a path. Type `cd /` in your unix console and you will get to the root directory. Exactly the same is true for all web servers. So you can tell that in the `http://example.com/` address the trailing slash is not for the decoration but a regular address itself - the address of the home page.

On Windows, the filesystem doesn't have the common root for the whole system but split between disks, so an absolute paths starts from the drive letter. Whereas each disk has its own root, which is a *backslash* - `\`. So you can type `cd \` and get to the root of the current disk.

So you can tell that windows is rather confusing, but for the simplicity we would pretend that we have only one disk, and within its boundaries the rules are pretty much the same as in Unix.

So now you can tell an absolute path from a relative one - it is starting from the root, which is:

- on a Unix file system it's `/`
- on a web server it's again `/`
- on Windows it's either `\` (for the current disk) or `D:\` (system-wide)

## Relative paths

If you don't supply the root, it means that your path is relative.

The simplest example of relative path is just a file name, like `index.html`. So one should be careful with relative paths. If your current directory is `/about/` then `index.html` would be one, but if you switch it to `/contacts/` then it will be another.

Other relative path examples:

- `./file.php` (the file is in the current folder. The same as just `file.php` )
- `images/picture.jpg` (the file is in the images folder that is in the current directory)
- `../file.php` (file is in the folder that is one level higher than the current directory)
- `../../file.php` (file is in the folder that is two levels higher than the current directory)

What you ought to know is that the system, when encountered a relative path, **always builds it up to the absolute one**. Both web-server and file system are doing that but different ways. So, let's learn them.

## Document root

This is the most interesting part. There is a point where the real world meets the virtual one.

Imagine there is a file like `/var/www/site/forum/index.php` . While on the web-server its address is `http://www.site.ru/forum/index.php`

And here the point can be clearly seen: there is a part, common for both addresses: `/forum/index.php` , which is the very source of confusion.

For the browser, this path is perfectly absolute, starting from the root of the web-server. Whereas for the script it's only a part of the full path - the filesystem path. And if you try to use it in PHP it will result in a failure: there is no `/forum/` catalog on the *HDD*!

To get the working path to this file, we have to add the missing part. In our example it's `/var/www/site` , which is called `DOCUMENT_ROOT` and is the most important configuration option for the file system interactions. In PHP you can access it via `$_SERVER['DOCUMENT_ROOT']` .

So now you can tell that to make **any** file system path work, it should be absolute and built using `DOCUMENT_ROOT` . So a correct PHP code to access `/forum/index.php` from PHP would be

```
$path = $_SERVER['DOCUMENT_ROOT'] . "/forum/index.php";
```

here we are using web-server part of the path, prepending it with the document root. Voila!

## Web server paths

are much simpler.

Like it was said before, for the browser, there are no files on the server. A site user never has an access to the server's file system. For the browser, there is a **site root** only. Which is constant *and always simply a slash*.

Therefore, to make an HTML link absolute, just build it from the site root - and you will never see a 404 error for the existing file again!

Imagine your site has two sections,

```
http://www.example.com/about/info.php  
and  
http://www.example.com/job/vacancy.php
```

and in the `info.php` you want to link to `vacancy.php`. If you make it as is, `<a href=vacancy.php>`, then browser won't find it! Remember, it always tries to build up the link to the full one, using the current location, which is `/about/` and so the resulting path is `/about/vacancy.php` which is wrong. To make it right, we have to make this link absolute, starting from the site root: `/job/vacancy.php`

So it goes for all the internal links on the site - images, js and css files, hyperlinks or any other resource that can be clicked on or loaded on the page.

For the local resources it's better to make it path only, without protocol and domain - like `/job/vacancy.php`. Whereas for the external resources these attributes are obligatory, and so it should be a fully qualified URL like `http://www.example.com/job/vacancy.php`.

## Console scripts. Single entry point

It's a pity, but for the console scripts our useful `$_SERVER['DOCUMENT_ROOT']` variable is unavailable. So we are bound to use paths relative to the calling script, derived from the current script's location.

For example, if your application is hosted in `/var/www/app` and there are two subfolders, `/var/www/app/bin` and `/var/www/app/config`, and you want to access the latter from the former, you can write the following code:

```
$config_path = __DIR__.'../config/settings.php';  
require $config_path;
```

Although technically absolute (starting from a slash), this path is essentially relative to the calling script, because if the calling script will be moved into another directory, it won't find the configuration file anymore.

This is why it is recommended to use a single entry point for your application. Or - as in our case - two entry points, one for web requests and one for console commands.

So for our fictional application we would have three files - an entry point for the web front, an entry point for console applications and a bootstrap file:

- `/var/www/app/html/index.php`
- `/var/www/app/bin/console.php`
- `/var/www/app/config/bootstrap.php`

Then we could write the following code (among other things) in `bootstrap.php`:

```
define('ROOT_DIR', realpath(__DIR__.'../'));
```

to define the `ROOT_DIR` constant that contains the path to our application's root directory (which is directly above the `config` dir).

And then in both `index.php` and `console.php` the

```
require __DIR__.'../config/bootstrap.php';
```

to make all the bootstrap stuff available, including the `ROOT_DIR` constant. From now on we can use it to build absolute paths starting from the root directory (as long as our scripts are called through the entry point either web or console one):

```
include ROOT_DIR.'/config/settings.php';
```

Example implementations can be found in Laravel's Artisan or Symfony Console.

Of course, both entry points should implement a sort of resolver to call all other pages and console scripts but that's slightly out of scope of this article.

## Helpful PHP commands and constants


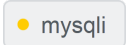
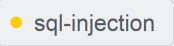
There are many helpful commands and constants in PHP to ease the path interpolation. Some of them are:

- `__FILE__` a constant that contains the full absolute path to the file which is currently executing.
- `__DIR__` a constant that contains the path to a directory where lies the file which is currently executing (effectively it's just `__FILE__` without the filename and the ending slash)
- `realpath()` command will convert a relative path to an absolute one
- `getcwd()` will give you the current directory

### RELATED ARTICLES:

- > Articles (/articles)
- > MVC in simpler terms or the structure of a modern web-application (/articles/mvc)
- > PHP error reporting (/articles/error\_reporting)
- > How to get a single player's rank based on the score (/articles/rank\_based\_on\_score)
- > Do you really need to check for both `isset()` and `empty()` at the same time? (/articles/empty)
- > Operator precedence or how does 'or die()' work. (/articles/or\_die)
- > Numerical strings comparison (/articles/numerical\_strings\_comparison)
- > What's wrong with popular articles telling you that foo is faster than bar? (/articles/single\_vs\_double)
- > Why should I use prepared statements if escaping is safe? (/articles/why\_should\_i\_use\_prepared\_statements\_if\_escaping\_is\_safe)
- > Do you abuse the null coalescing operator (and `isset/empty` as well)? (/articles/null\_coalescing\_abuse)

### GOT A QUESTION?

I am the only person to hold a gold badge in  <http://stackoverflow.com/help/badges/4220/pdo>,  <http://stackoverflow.com/help/badges/6342/mysqli> and 

(<http://stackoverflow.com/help/badges/5981/sql-injection>) on Stack Overflow and I am eager to show the right way for PHP developers.

Besides, your questions let me make my articles even better, so you are more than welcome to ask any question you got.

[Click here to ask!](#)

## LATEST ARTICLE:

[PHP Error Reporting \(/articles/error\\_reporting\)](/articles/error_reporting)

## SEE ALSO:

- > [Top 10 PHP delusions \(/top\)](/top)
- > [PDO Examples \(/pdo\\_examples\)](/pdo_examples)
- > [Mysqli Examples \(/mysqli\\_examples\)](/mysqli_examples)
- > [Articles \(/articles\)](/articles)
- > [MVC in simpler terms or the structure of a modern web-application \(/articles/mvc\)](/articles/mvc)
- > [PHP error reporting \(/articles/error\\_reporting\)](/articles/error_reporting)
- > [How to get a single player's rank based on the score \(/articles/rank\\_based\\_on\\_score\)](/articles/rank_based_on_score)
- > [Do you really need to check for both isset\(\) and empty\(\) at the same time? \(/articles/empty\)](/articles/empty)
- > [Operator precedence or how does 'or die\(\)' work. \(/articles/or\\_die\)](/articles/or_die)
- > [Numerical strings comparison \(/articles/numerical\\_strings\\_comparison\)](/articles/numerical_strings_comparison)
- > [What's wrong with popular articles telling you that foo is faster than bar? \(/articles/single\\_vs\\_double\)](/articles/single_vs_double)
- > [Why should I use prepared statements if escaping is safe? \(/articles/why\\_should\\_i\\_use\\_prepared\\_statements\\_if\\_escaping\\_is\\_safe\)](/articles/why_should_i_use_prepared_statements_if_escaping_is_safe)
- > [Do you abuse the null coalescing operator \(and isset/empty as well\)? \(/articles/null\\_coalescing\\_abuse\)](/articles/null_coalescing_abuse)

## LATEST COMMENTS:

18.04.20 00:46

**Lonestar Jack** for [How to connect to MySQL using PDO:](#)

[\(/pdo\\_examples/connect\\_to\\_mysql#comment-846\)](/pdo_examples/connect_to_mysql#comment-846)

What is the best coding to use two queries on one page? One connection and then close it and do...

[read more \(/pdo\\_examples/connect\\_to\\_mysql#comment-846\)](#)

17.04.20 19:02

**Gregory Stathes** for [How to use mysqli properly: \(/mysqli#comment-845\)](#)

My issue is this, been trying to solve for days. And I am more than happy to pay for a solution,...  
read more (/mysql/#comment-845)

16.04.20 20:47

**Phil** for How to use mysqli properly: (/mysqli#comment-844)

## Which is better for performance mysqli or PDO?

[read more \(/mysqli#comment-844\)](#)

12.04.20 04:28

**eljaydee** for Mysqli examples: (/mysqli\_examples#comment-843)

I cannot make this work: I have been told a million times to "review you query" but cannot find...

[read more \(/mysqli\\_examples#comment-843\)](#)

12.04.20 04:27

**Elizabeth** for PDO Examples: (/pdo\_examples#comment-842)

Hi, my php programs which worked fine on my local server are now no longer passing through...

[read more \(/pdo\\_examples#comment-842\)](#)

Tweets by @ShrapnelCol (<https://twitter.com/ShrapnelCol>)

## Add a comment

Please refrain from sending spam or advertising of any sort.

Messages with hyperlinks will be pending for moderator's review.

## Markdown is now supported:

- > before and an empty line after for a quote
- four spaces to mark a block of code

Your name

Are you a robot?

## Message

Address

If you want to get a reply from admin, you may enter your E—mail address above

Send

## Comments:

>, 10.03.20 15:36

ALAMIN JUMA, 18.02.20 18:45

Hey, I want to access a file located in a different folder. The file is called config.php and contains database configurations and its located in libraries folder. When i require it in another file called tables.php inside admins folder it gives me a big error "require not found blablablaa". Help!

REPLY:

Hello Alamin

Just read the article above, it explains what to do. Basically you need to use an absolute path and use `$_SERVER[ 'DOCUMENT_ROOT' ]` for it

Mark, 01.05.19 23:10

Thank you very much. I am trying to create a secure site where my HTML/CSS/JS files are in the webserver root and all my PHP files are in the filesystem root (I think I have that correct) and therefore not visible to prying eyes. I'll be studying this page for a while until I get it right.

REPLY:

Hello Mark!



Thank you very much for your kind words.

Please don't hesitate to ask if you have any questions. Bear in mind that this way you will help me to make the article better, hence all questions are welcome!

Sam, 09.01.19 13:20

You should write a book. Seriously. Very well explained and extremely helpful. Thank You for sharing your knowledge.

rob, 03.10.18 00:09

it's great but unfortunately when the script is executed via cron it doesn't see \$\_SERVER because it's not set. What would you recommend as a work around?

Thank you

REPLY:

Hello Rob!

Thank you for the very good question! Indeed, this is the very probable and very important case.

I added another section to the article, in this exact case. Hope it would serve you well:

<https://phpdelusions.net/articles/paths#console> (<https://phpdelusions.net/articles/paths#console>)

If the answer would not satisfy you, please share your doubts. Thank you for helping me to make the site better!

Srinidhi, 18.09.18 09:12

Hi, We have an application hosted in IIS. This application has file upload control. When i chose a file to upload from a folder it takes the full path along with file name and this full path is considered as file name in the system. Some times it exceeds max file name length and throws an error. This feature was working file before. But now we are migrating to another server. Are we missing any IIS feature in this new Server so that it is talking full path instead of just file name? Please Help

REPLY:

Hello Srinidhi!

According to my experience, the full path is sent when the form tag doesn't have the "multipart/form-data" attribute. But I don't have much experience with IIS.

bhavin, 08.08.18 13:11

Very informative article.

Millar, 05.06.18 16:02

Agreed. Good article. Having someone explain that the base directory is just a / is a real help, as most people tend to skip over that.

David, 12.02.18 04:40

Very helpful article. Relative and absolute paths kept throwing me off. Seems each time I had to code something that would save a user-uploaded file to the server, I'd be trying a few different folder paths until I got one that worked and call it a day, but not really understanding if the code I used was the optimal, robust version.

About (/about) © phpdelusions.net, 2020