

PHP (http://www.sitepoint.com/php/)

Working with Files in PHP



[\(http://www.sitepoint.com/author/itench/\)](http://www.sitepoint.com/author/itench/)

Iain Tench (<http://www.sitepoint.com/author/itench/>)

Published April 23, 2012

 **Tweet** (<https://twitter.com/share?text=Working+with+Files+in+PHP&via=sitepointdotcom>)

Subscribe (<https://confirmsubscription.com/h/y/D46DD8C9BD8A0B5D>)

You may well be familiar with databases such as MySQL and Access which are an ever-increasingly common means of storing data. But data is also stored in files, like Word documents, event logs, spreadsheets, image files, and so on. Databases generally require a special query language to retrieve information, whereas files are ‘flat’ and usually appear as a stream of text.

Most often when working with files you’ll read from or write to them. When you want to read the contents of a file you first have to open it, then read as much of the contents as you want, then close the file when you’re finished. When writing to a file, it also needs to be opened (or perhaps created if it doesn’t already exist), then you write your data to it, and close the file when you have finished. In PHP5 there are some built-in wrapper functions that handle the opening and closing automatically for you, but it still happens under the hood.

You may also find it useful to find out more about the file by examining its attributes before you start working with it. For example, does the file exist? When was it last updated? When was it created?

PHP provides a range of functions which allow you to work with files, and in this article I’ll demonstrate some of them for you.

File Attributes

File attributes are the properties of a file, for example its size, the last time it was accessed, its owner, etc. Let's look at how you find out more about the files you're working with.

File Size

The function `filesize()` retrieves the size of the file in bytes.

```
1 <?php
2 $f = "C:\Windows\win.ini";
3 $size = filesize($f);
4 echo $f . " is " . $size . " bytes.";
```

When executed, the example code displays:

```
C:Windows\win.ini is 510 bytes.
```

The use of a file on a Windows system here highlights an important point; because the backslash has special meaning as an escape character in strings, you'll need to escape it by adding another backslash.

If the file doesn't exist, the `filesize()` function will return false and emit an `E_WARNING`, so it's better to check first whether the file exists using the function `file_exists()`:

```
1 <?php
2 $f = "C:\Windows\win.ini";
3 if (file_exists($f)) {
4     $size = filesize($f);
5     echo $f . " is " . $size . " bytes.";
6 }
7 else {
8     echo $f . " does not exist.";
9 }
```

In fact many of the functions presented in this section have the same behavior, i.e., emit warnings. I've not included a check with `file_exists()` in the rest of my examples for the sake of brevity, but you'll want to use it when you write your own code.

File History

To determine when a file was last accessed, modified, or changed, you can use the following functions respectively: `fileatime()` , `filemtime()` , and `filectime()` .

```
1 <?php
2 $dateFormat = "D d M Y g:i A";
3
4 $atime = fileatime($f);
5 $mtime = filemtime($f);
6 $ctime = filectime($f);
7
8 echo $f . " was accessed on " . date($dateFormat, $atime) . "<br>";
9 echo $f . " was modified on " . date($dateFormat, $mtime) . "<br>";
10 echo $f . " was changed on " . date($dateFormat, $ctime) . " .";
```

The code here retrieves the timestamp of the last access, modify, and change dates and displays them,

```
C:Windowswin.ini was accessed on Tue 14 Jul 2009 4:34 AM.
C:Windowswin.ini was modified on Fri 08 Jul 2011 2:03 PM.
C:Windowswin.ini was changed on Tue 14 Jul 2009 4:34 AM.
```

To clarify, `filemtime()` returns the time when the contents of the file was last modified, and `filectime()` returns the time when information associated with the file, such as access permissions or file ownership, was changed.

The `date()` function was used to format the Unix timestamp returned by the `file*time()` functions. Refer to the [documentation for the `date\(\)` function](http://php.net/manual/en/function.date.php) (<http://php.net/manual/en/function.date.php>) for more formatting options.

File Permissions

Before working with a file you may want to check whether it is readable or writeable to the process. For this you'll use the functions `is_readable()` and `is_writeable()` :

```
1 <?php
2 echo $f . (is_readable($f) ? " is" : " is not") . " readable.<br>";
3 echo $f . (is_writable($f) ? " is" : " is not") . " writeable.";
```

Both functions return a Boolean value whether the operation can be performed on the file. Using the ternary operator you can tailor the display to state whether the file is or is not accessible as appropriate.

```
C:Windows\win.ini is readable.  
C:Windows\win.ini is not writeable.
```

File or Not?

To make absolutely sure that you're dealing with a file you can use the `is_file()` function. `is_dir()` is the counterpart to check if it is a directory.

```
1 <?php  
2 echo $f . (is_file($f) ? " is" : " is not") . " a file.<br>";  
3 echo $f . (is_dir($f) ? " is" : " is not") . " a directory.";
```

The example code outputs:

```
C:Windows\win.ini is a file.  
C:Windows\win.ini is not a directory.
```

Reading Files

The previous section showed how you can find out a good deal about the files you're working with before you start reading or writing to them. Now let's look at how you can read the contents of a file.

The convenience function `file_get_contents()` will read the entire contents of a file into a variable without the need to open or close the file yourself. This is handy when the file is relatively small, as you wouldn't want to read in 1GB of data into memory all at once!

```
1 <?php  
2 $f = "c:\windows\win.ini";  
3 $f1 = file_get_contents($f);  
4 echo $f1;
```

For larger files, or just depending on the needs of your script, it's may be wiser to handle the details

yourself. This is because once a file is opened you can seek to a specific offset within it and read as little or as much data as you want at a time.

The function `fopen()` is used to open the file:

```
1 <?php
2 $f = "c:\windows\win.ini";
3 $f1 = fopen($f, "r");
```

Using the function `fopen()`, two arguments are needed – the file I want to open and the mode, which in this case is “r” for read. The function returns a handle or stream to the file, stored in the variable `$f1`, which you use in all subsequent commands when working with the file.

The most common mode values are:

| Mode | Meaning | File pointer position | What happens to contents? | If the file doesn't exist? |
|------|------------|-----------------------|---------------------------|----------------------------|
| r | Read only | Beginning of file | Unchanged | Warning message |
| w | Write only | Beginning of file | Existing contents erased | Will try to create |
| a | Write only | End of file | Add to existing contents | Will try to create |

For other values, refer to the list in [PHP's `fopen\(\)` page](http://php.net/manual/en/function.fopen.php) (<http://php.net/manual/en/function.fopen.php>).

To read from the opened file one line at a time, use the function `fgets()`:

```
1 <?php
2 $f = "c:\windows\win.ini";
3 $f1 = fopen($f, "r");
4 do {
5     echo fgets($f1) . "<br>";
6 }
7 while (!feof($f1));
8 fclose($f1);
```

Using a `do - while` loop is a good option because you may not know in advance how many lines are in the file. The function `feof()` checks whether the file has reached its end – the loop continues until the end of file condition is reached.

To tidy up after I've finished reading the file, the function `fclose()` is used to close the file.

Writing Files

Two commonly used modes when writing to a file using the function `fwrite()` are “w” and “a” – “w” indicates you want to write to the file but it will remove any of the existing file contents beforehand, while “a” means that you will append any new data to what already exists in the file. You need to be sure you are using the correct option!

In this example I’ll use “a” to append:

```
1 <?php
2 $file = "add_emp.txt";
3 $f1 = fopen($file, "a");
4 $output = "banana" . PHP_EOL;
5 fwrite($f1, $output);
6 $output = "cheese" . PHP_EOL;
7 fwrite($f1, $output);
8 fclose($f1);
```

First the file name is assigned to a variable, then the file is opened in mode “a” for append. The data to be written is assigned to a variable, `$output`, and `fwrite()` adds the data to the file. The process is repeated to add another line, then the file is closed using `fclose()`. The pre-defined constant `PHP_EOL` adds the newline character specific to the platform PHP is running on.

The file contents after executing the above code should look like this:

```
banana
cheese
```

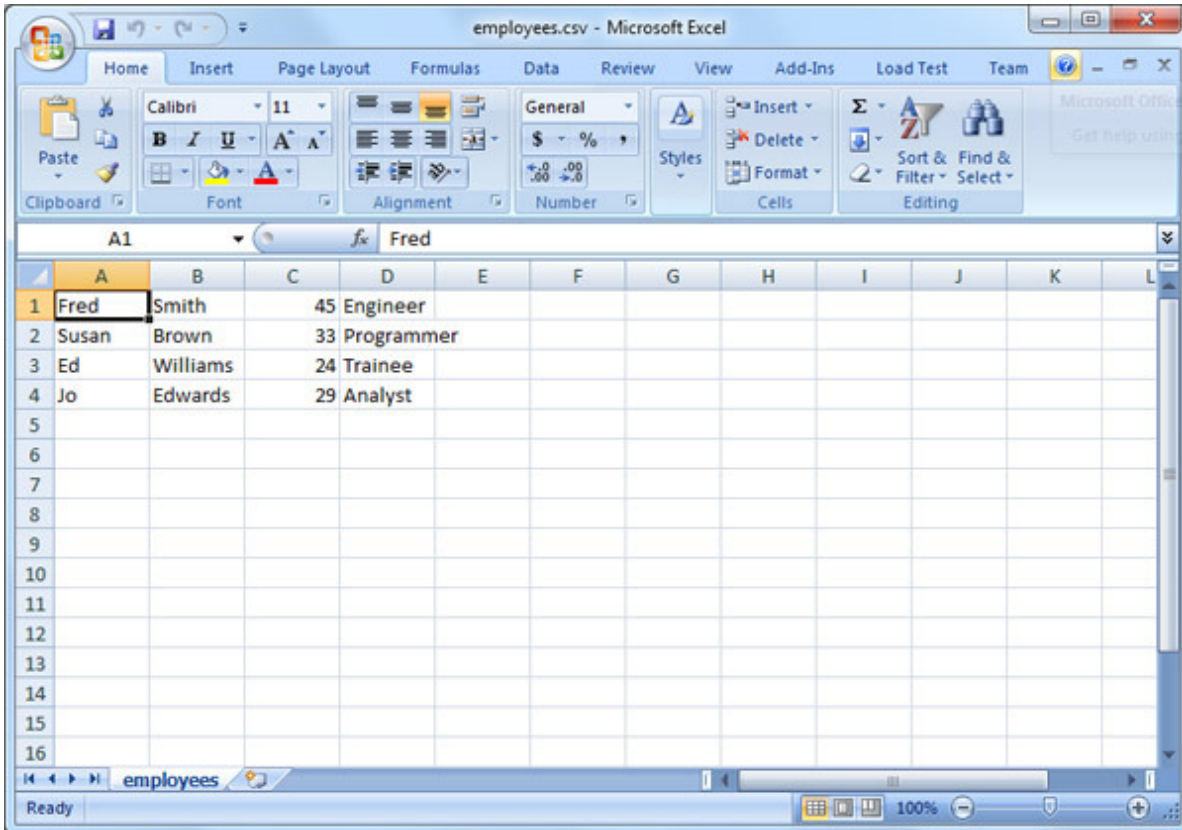
The convenience function `file_put_contents()` can also write to a file. It accepts the file name, the data to be written to the file, and the constant `FILE_APPEND` if it should append the data (it will overwrite the file’s contents by default).

Here’s the same example as above, but this time using `file_put_contents()`:

```
1 <?php
2 $file = "add_emp.txt";
3 file_put_contents($file, "banana" . PHP_EOL);
4 file_put_contents($file, "cheese" . PHP_EOL, FILE_APPEND);
```

Working with CSV Files

CSV stands for comma separated variable and indicates the file contains data delimited with a comma. Each line is a record, and each record is made up of fields, very much like a spreadsheet. In fact, software such as Excel provides the means to save files in this format. Here is an example as displayed in Excel 2007 (which doesn't show the commas):



| | | | | | | | | | | | | |
|----|-------|----------|----|------------|---|---|---|---|---|---|---|---|
| | A | B | C | D | E | F | G | H | I | J | K | L |
| 1 | Fred | Smith | 45 | Engineer | | | | | | | | |
| 2 | Susan | Brown | 33 | Programmer | | | | | | | | |
| 3 | Ed | Williams | 24 | Trainee | | | | | | | | |
| 4 | Jo | Edwards | 29 | Analyst | | | | | | | | |
| 5 | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | |
| 10 | | | | | | | | | | | | |
| 11 | | | | | | | | | | | | |
| 12 | | | | | | | | | | | | |
| 13 | | | | | | | | | | | | |
| 14 | | | | | | | | | | | | |
| 15 | | | | | | | | | | | | |
| 16 | | | | | | | | | | | | |

There are 4 lines of data, or records, in the file containing first name, last name, age and job title. To read the file and extract the individual fields, start by opening the file in read mode:

```
1 <?php
2 $file = "employees.csv";
3 $f = fopen($file, "r");
```

Now I need to use a function specifically to read the CSV formatted file, `fgetcsv()` ; a while loop is used to cater for the fact that the number of records is expected to vary:

```
1 <?php
2 $file = "employees.csv";
3 $f = fopen($file, "r");
4 while ($record = fgetcsv($f)) {
5     foreach($record as $field) {
6         echo $field . "<br>";
7     }
8 }
9 fclose($f);
```

This will produce output with a field on each line:

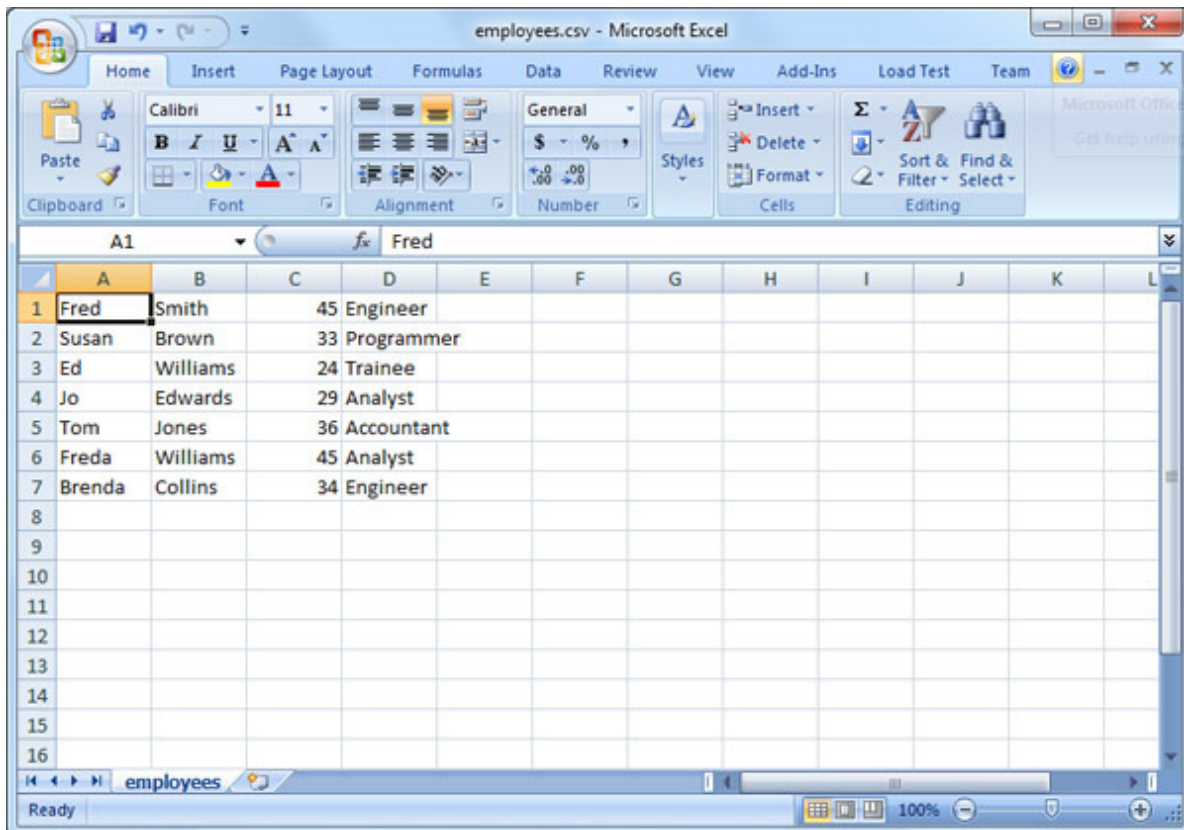
```
Fred
Smith
45
Engineer
Susan
Brown
33
Programmer
Ed
Williams
24
Trainee
Jo
Edwards
29
Analyst
```

Let's examine the loop. First, `fgetcsv()` has one argument only – the file handle; the comma delimiter is assumed. `fgetcsv()` returns data in an array which is then processed using a `foreach` loop. When all records have been read the file is closed using `fclose()`.

Let's now look at how you can update or write to a CSV file, this time using the function `fputcsv()`. Because I am using the same file that I used above, I will open it in append mode, and the data to be added to the file has been defined in an array.


```
1 <?php
2 $file = "employees.csv";
3 $f = fopen($file, "a");
4
5 $newFields = array(
6     array('Tom', 'Jones', 36, 'Accountant'),
7     array('Freda', 'Williams', 45, 'Analyst'),
8     array('Brenda', 'Collins', 34, 'Engineer'));
9
10 foreach($newFields as $fields) {
11     fputcsv($f, $fields);
12 }
13 fclose($f);
```

The contents of the file as displayed in Excel now look like this:



| | A | B | C | D | E | F | G | H | I | J | K | L |
|----|--------|----------|----|------------|---|---|---|---|---|---|---|---|
| 1 | Fred | Smith | 45 | Engineer | | | | | | | | |
| 2 | Susan | Brown | 33 | Programmer | | | | | | | | |
| 3 | Ed | Williams | 24 | Trainee | | | | | | | | |
| 4 | Jo | Edwards | 29 | Analyst | | | | | | | | |
| 5 | Tom | Jones | 36 | Accountant | | | | | | | | |
| 6 | Freda | Williams | 45 | Analyst | | | | | | | | |
| 7 | Brenda | Collins | 34 | Engineer | | | | | | | | |
| 8 | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | |
| 10 | | | | | | | | | | | | |
| 11 | | | | | | | | | | | | |
| 12 | | | | | | | | | | | | |
| 13 | | | | | | | | | | | | |
| 14 | | | | | | | | | | | | |
| 15 | | | | | | | | | | | | |
| 16 | | | | | | | | | | | | |

Summary

This article has given you an introduction into working with files and shown how you can find information about files, read from and write to files. There are many other functions you can use when working with files – this article has covered some of the most commonly used. You'll find more information on PHP's [file](#)

[system functions page \(http://php.net/manual/en/ref.filesystem.php\)](http://php.net/manual/en/ref.filesystem.php).

Image via [Marco Rullkoetter \(http://www.shutterstock.com/gallery-172795p1.html\)](http://www.shutterstock.com/gallery-172795p1.html) / Shutterstock (<http://www.shutterstock.com>)



(<http://www.sitepoint.com/author/itench/>)

[Iain Tench \(http://www.sitepoint.com/author/itench/\)](http://www.sitepoint.com/author/itench/)

Iain Tench has worked in the IT industry for 30 years as a programmer, project manager (Prince2 Practitioner), consultant, and teacher. As a project manager, he specialized in integration projects mainly involving payment systems in the financial sector. He now teaches and has acquired a Masters degree in Internet Systems Development as well as a teaching qualification. Iain's specialized areas of teaching is web technologies, mainly programming in a variety of languages, HTML, CSS and database integration. He's also written a book about Dreamweaver CS3 for Hodder Education.

Related Articles

[The Pros and Cons of Zend Certification \(http://www.sitepoint.com/pros-cons-zend-certification/\)](http://www.sitepoint.com/pros-cons-zend-certification/) ➤

[HHVM revisited \(http://www.sitepoint.com/hhvm-revisited/\)](http://www.sitepoint.com/hhvm-revisited/) ➤

[Setting Up PHP behind Nginx with FastCGI \(http://www.sitepoint.com/setting-up-php-behind-nginx-with-fastcgi/\)](http://www.sitepoint.com/setting-up-php-behind-nginx-with-fastcgi/) ➤



Free book: Jump Start HTML5 Basics

Grab a free copy of one our latest ebooks! Packed with hints and tips on HTML5's most powerful new features.

Claim Book

Comments for this thread are now closed.

13 Comments

SitePoint

 Login

Sort by Best

Share  Favorite 



Kiash • a year ago

Is there any way to working with DOC file using PHP?

• Share ›



Kedar • 2 years ago

What about File Uploading?

• Share ›



nice work! • 2 years ago

but i can open only windows directory, what to do for reading other file of other directory

• Share ›



asad_pk • 2 years ago

Working with CSV Files example is very nice .Thanks

• Share ›



ChonUnca • 2 years ago

would be more interesting if the "write file" section talk about offset, because it's the most complicated part and the less documented. How to insert a line between 2 others, or insert some chars in a line? I'd hair teared my self for a long time before understood how it worked.

• Share ›



Mikael Randy • 2 years ago

We are in 2012, PHP current version is 5.4.

From PHP 5.1 (2005 ! 7 years ago !), we ca use the incredible SPL (<http://www.php.net/manual/en/b...> and especially files handling classes (<http://fr.php.net/manual/en/sp....>

But what the hell with theses developers who continue to use these functions from another time?

• Share ›



overcooked ➔ Mikael Randy • 2 years ago

For anyone interested in SPL, a good introduction:

<http://www.phpro.org/tutorials...>

• Share ›



Jason Knighth ➔ Mikael Randy • 2 years ago

Oh, you want to see just how malfing stupid the SPL documentation is? Look no further than 'appendIterator'.

... and I quote:

"An Iterator that iterates over several iterators one after the other."

About

[About us \(/about-us/\)](/about-us/)

[Advertise \(/advertising\)](/advertising/)

[Legals \(/legals\)](/legals/)

[Feedback \(mailto:feedback@sitepoint.com\)](mailto:feedback@sitepoint.com)

Our Sites

[Learnable \(https://learnable.com\)](https://learnable.com)

[Reference \(http://reference.sitepoint.com\)](http://reference.sitepoint.com)

[Hosting Reviews \(/hosting-reviews/\)](/hosting-reviews/)

[Web Foundations \(/web-foundations/\)](/web-foundations/)

Connect



[\(/feed\)](/feed/) [\(/newsletter\)](/newsletter/) [\(/https://www.facebook.com/sitepoint\)](https://www.facebook.com/sitepoint)



[\(/http://twitter.com/sitepointdotcom\)](http://twitter.com/sitepointdotcom) [\(/https://plus.google.com/+sitepoint\)](https://plus.google.com/+sitepoint)

© 2000 – 2014 SitePoint Pty. Ltd.