File System Functions

Disclaimer: вы смотрите просто запись лекции,
это **НЕ** специально подготовленный видеокурс!

Intro

File system operations are everywhere. No mater what you are creating – CLI utility, web CMS, etc. – you have to deal with file operations on daily basis.

Full list of functions: https://www.php.net/manual/en/refs.fileprocess.file.php

Warning! When dealing with file systems, mind OS specifics, directory separators, access permissions and so on!

Important constants

To make your PHP application platform-independent you have to use the following predefined constants:

```php
<?php

echo  DIRECTORY_SEPARATOR . "\n";
// On Windows: \
// On *nix: /

echo PATH_SEPARATOR . "\n";
// On Windows: ;
// On *nix: :
```

Checking for an object existence

To check if a file system object exists (regardless of its type) wey ma use the following function:

```php
<?php

if (file_exists('c:/dir1/dir2/some_file.txt')) {
    echo 'Yes';
} else {
    echo 'No';
}
```

This path may lead to ANY file system object (not just files).

Also note that ‘/’ directory separator works on Windows and other OSes, but ‘\’ works on Windows ONLY!

Checking an object type

To check a file system object type (along with its existence) we may use the following set of functions:

```
<?php

var_dump(is_file('C:/Windows/notepad.exe')); // bool(true)
var_dump(is_dir('c:/')); // bool(true)
var_dump(is_link('C:/Windows/notepad.exe')); // bool(false)
```

Checking timestamp of an object

To check a timestamp of file system object we may use the following set of functions:

Some file systems support only subset of these timestamps!

```php
<?php

echo filectime('C:/Windows/notepad.exe') . "\n"; // 1636203359 -- change time
echo filemtime('C:/Windows/notepad.exe') . "\n"; // 1636203359 -- modification time
echo fileatime('C:/Windows/notepad.exe') . "\n"; // 1646833333 -- last access time

var_dump(stat('C:/Windows/notepad.exe'));
```

This function returns huge amount of data. See the official PHP documentation for details.

```
/*
See https://www.php.net/manual/en/function.stat.php for details:
array(26) {
  [0] => int(4208659137)
  [1] => int(281474977580420)
  [2] => int(33279)
  [3] => int(3)
  [4] => int(0)
  [5] => int(0)
  [6] => int(0)
  [7] => int(208384)
  [8] => int(1646833333)
  [9] => int(1636203359)
  [10] => int(1636203359)
  [11] => int(-1)
  [12] => int(-1)
  'dev' => int(4208659137)
  'ino' => int(281474977580420)
  'mode' => int(33279)
  'nlink' => int(3)
  'uid' => int(0)
  'gid' => int(0)
  'rdev' => int(0)
  'size' => int(208384)
  'atime' => int(1646833333)
  'mtime' => int(1636203359)
  'ctime' => int(1636203359)
  'blksize' => int(-1)
  'blocks' => int(-1)
}
*/
```

Checking size of an object

To check a file system object size we may use two approaches depending on the object type:

```php
<?php

echo filesize('C:/Windows/notepad.exe'). "\n"; // 208384
```

The filesize() function can NOT calculate the real directory size, we need a special approach for directories… (See the next slide.)

Checking size of an object

To check a file system object size we may use two approaches depending on the object type:

```php
<?php

function getDirectorySize(string $directoryToScan) : int
{
    if (!is_dir($directoryToScan)) {
        throw new Exception('getDirectorySize expects its parameter to be a valid directory path.');
    }

    $totalSize = 0;
    $directoryDescriptor = opendir($directoryToScan);
    if ($directoryDescriptor === false) {
        return 0;
        // Or you may throw an exception instead:
        // throw new Exception('Failed to open [' . $directoryToScan . '].');
    }

    while (false !== ($objectName = readdir($directoryDescriptor))) {
        if (($objectName === '.') || ($objectName === '..')) {
            continue;
        }
        $fullName = realpath($directoryToScan . DIRECTORY_SEPARATOR . $objectName);
        if (is_dir($fullName)) {
            $totalSize += getDirectorySize($fullName);
        } else {
            $totalSize += filesize($fullName);
        }
    }

    closedir($directoryDescriptor);
    return $totalSize;
}

echo getDirectorySize('C:/Program Files'); // 22156607810
```

To calculate the real directory size we need a special approach, i.e. we need a recursive function.

Reading a text file into an array of strings

To read a text file into and array of strings (lines) we may use the following function:

```php
<?php

print_r(file('C:/Windows/System32/drivers/etc/hosts'));
/*
Array
(
    [0] => # Copyright (c) 1993-2009 Microsoft Corp.
    [1] => #
    [2] => # This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
    [3] => #
    [4] => # This file contains the mappings of IP addresses to host names. Each
    [5] => # entry should be kept on an individual line. The IP address should
    ...
)
*/
```

Reading and writing any files

To read/write data from/to any file (text or binary) we may use the following functions:

```
<?php

// To read the whole file at once:
$fileData = file_get_contents('C:/Windows/System32/drivers/etc/hosts');

// To read a part of a file:
$partialFileData = file_get_contents('C:/Windows/System32/drivers/etc/hosts', false, null, 100, 50);

// To overwrite a file:
file_put_contents('test.txt', $fileData);

// To append a file:
file_put_contents('test.txt', "\nNew line...", FILE_APPEND);
```

Always use **false** here, otherwise you risk to access a wrong file.

Reading and writing any files the old-school way

To read/write data from/to any file (text or binary) we may also use the following functions:

```php
<?php

$inputFileResource = fopen("file1.dat", "rb");
$outputFileResource = fopen("file2.dat", "wb");

while (!feof($inputFileResource)) {
    $data = fread($inputFileResource, 2048);
    fwrite($outputFileResource, $data);
}

fclose($inputFileResource);
fclose($outputFileResource);
```

This approach literally works the same way as in C/C++.

How to copy, rename (move) or delete a file

To copy, rename (move) or delete a file we may use the following functions (these functions are for FILES, not for directories):

```
<?php

// This function NEVER works for directories:
copy("c:/1.txt","d:/2.txt");

// This function MAY work for directories in rare cases:
rename("c:/11.txt","d:/22.txt");

// This function MAY work for directories in rare cases:
unlink("/home/dir/file.ext");
```

To copy/rename/delete a directory we usually need a complex recursive approach.

Scanning a directory

There are several ways to scan a directory in PHP, yet the most universal is the second one:

```
<?php

print_r(scandir('.'));

/*
Array
(
    [0] => .
    [1] => ..
    [2] => 01_predefined_constants.php
    ...
)
*/
```

This approach only provides us with a list of NAMES of file system objects. Sometimes that’s enough, but usually it’s not.

```
<?php

$directoryDescriptor = opendir('.');
while (false !== ($objectName = readdir($directoryDescriptor))) {
    echo $objectName . ' = ' . filesize($objectName) . "\n";
}
closedir($directoryDescriptor);

/*
. = 4096
.. = 4096
01_predefined_constants.php = 139
...
*/
```

This approach allows us to collect any information about any file system object and/or to perform any operation with that object.

Typical directory operations

While most directory operations require complex recursive approach, there are some simple operations available:

```php
<?php

// Getting and changing CWD (current working directory):
echo getcwd() . "\n"; // D:\PWD_Code_Samples\05 05 - File System
Functions
chdir('c:/');
echo getcwd() . "\n"; // C:\

// Creating and deleting a directory:
mkdir('c:/1');
rmdir('c:/1');
```

Extracting file name parts

To extract any part from a file name we may (and should!) use the following function:

```
<?php

print_r(pathinfo('c:/dir1/dir2/file.ext'));

/*
Array
(
     [dirname] => c:/dir1/dir2
     [basename] => file.ext
     [extension] => ext
     [filename] => file
)
*/
```

Please stop “reinventing the wheel” with ugly string expressions to detect a file extension or some other path of a file name.

P.S. Everything is a file…

Most PHP file functions may work with a lot of wrappers. See https://www.php.net/manual/en/wrappers.php for details:

```php
<?php

print_r(file('http://svyatoslav.biz'));
/*
Array
(
    [0] => <!DOCTYPE html>
    [1] => <!--[if IE 7]>
    [2] => <html class="ie ie7" lang="ru-RU">
     ...
)
*/

$archivedFileDescriptor = fopen('zip://./archive.zip#file.txt', 'r');
if ($archivedFileDescriptor) {
    while (!feof($archivedFileDescriptor)) {
        echo fread($archivedFileDescriptor, 8192);
    }
    fclose($archivedFileDescriptor);
}
```

P.P.S. Everything REALLY is a file…

Here are just a quick samples on some standard wrappers usage. This is unconventional, but still useful in some cases.
See details: https://www.php.net/manual/en/wrappers.php.php

```php
<?php

echo "Enter something (or press ENTER to exit): \n";
$stdIn = fopen('php://stdin', 'r');
$stdOut = fopen('php://stdout', 'w');
$stdErr = fopen('php://stderr', 'w');
while ($line = trim(fgets($stdIn))) {
    if ($line == '') {
        break;
    }
    fputs($stdOut, 'You\'ve entered: ' . $line . "\n");
    fputs($stdErr, 'Not an error, just for fun: ' . $line . "\n");
}
fclose($stdIn);
echo "OK, exiting...";
```

There are more I/O wrappers. These are just the simplest ones.

File System Functions

Disclaimer: вы смотрите просто запись лекции,
это **НЕ** специально подготовленный видеокурс!