PHP – System Calls

PHP's library of built-in function includes a category of functions that deal with invoking operating system utilities and external programs from within the PHP code. In this chapter, we shall discuss the PHP functions used to perform system calls.

The system() Function

The system() function is similar to the system() function in C that it executes the given command and outputs the result.

```
system(string $command, int &$result_code = null): string|false
```

The system() call tries to automatically flush the web server's output buffer after each line of output if PHP is running as a server module. It returns the last line of the command output on success, and false on failure.

Example

The following PHP snippet invokes DIR command of Windows OS and displays the list of files in the current directory.

```
<?php
   echo '<pre>';

   // Outputs all the result of DOS command "dir", and returns
   // the last output line into $last_line. Stores the return value
   // of the shell command in $retval.
   $last_line = system('dir/w', $retval);

   // Printing additional info
   echo '
   </pre>
   <hr />Last line of the output: ' . $last_line . '
   <hr />Return value: ' . $retval;
?>
```

It will produce the following **output** –

```
Volume in drive C has no label.
Volume Serial Number is 7EE4-E492

Directory of C:\xampp\htdocs
[.]                 [..]                applications.html   bitnami.css
[dashboard]         employee.csv        favicon.ico         hello.csv
hello.html          hello.php           homepage.php        [img]
index.php           [Langi]             menu.php            myform.php
myname.php          new.png             new.txt             test.php
test.zip            [TPcodes]           uploadfile.php      [webalizer]
welcome.png         [xampp]
                18 File(s)        123,694 bytes
                8 Dir(s)          168,514,232,320 bytes free

Last line of the output: 8 Dir(s) 168,514,232,320 bytes free
Return value: 0
```

The shell_exec() Function

The shell_exec() function is identical to PHP's backtick operator. It executes the given command via shell and return the complete output as a string

```
shell_exec(string $command): string|false|null
```

The function returns a string containing the output from the executed command, false if the pipe cannot be established or null if an error occurs or the command produces no output.

Example

In the following code, we use shell_exec() function to obtain a list of files with ".php" as the extension in the current directory −

```
<?php
   $output = shell_exec('dir *.php');
   echo "<pre>$output</pre>";
?>
```

It will produce the following **output** −

```
Volume in drive C has no label.
Volume Serial Number is 7EE4-E492

Directory of C:\xampp\htdocs

10/26/2023  08:27 PM                73 hello.php
10/12/2023  10:40 AM                61 homepage.php
07/16/2015  09:02 PM               260 index.php
10/12/2023  10:39 AM                49 menu.php
09/25/2023  01:43 PM               338 myform.php
10/12/2023  10:49 AM                51 myname.php
10/26/2023  02:00 PM               369 test.php
09/25/2023  01:42 PM               555 uploadfile.php
               8 File(s)          1,756 bytes
               0 Dir(s)   168,517,771,264 bytes free
```

The exec() Function

The exec() function executes the given command as a string argument.

```
exec(string $command, array &$output = null,
   int &$result_code = null):string|false
```

The **$output** parameter, if specified, is an array that will be filled with every line of output from the command.

Example

In this case, we use exec() function to call whoami command from inside the program. The whoami command returns the username.

```
<?php

   // outputs the username that owns the running php/httpd process
   // (on a system with the "whoami" executable in the path)
   $output=null;
   $retval=null;
   exec('whoami', $output, $retval);
   echo "Returned with status $retval and output:\n";
   var_dump($output);

?>
```

It will produce the following **output** −

```
Returned with status 0 and output: array(1)
{ [0]=> string(13) "gnvbgl3\mlath" }
```

The passthru() Function

The passthru() function executes an external program and display raw output. Though the passthru() function is similar to the exec() or system() function in that it executes a command, it should be used in their place when the output from the OS command is binary data which needs to be passed directly back to the browser.

Example

A PHP program that uses passthu() function to display the contents of system PATH environment variable

```
passthru(string $command, int &$result_code = null): ?false
<?php
   passthru ('PATH');
?>
```

It will produce the following **output** −

```
PATH=C:\Python311\Scripts\;C:\Python311\;C:\WINDOWS\system32;C:\WINDOWS;
C:\WINDOWS\System32\Wbem;C:\WINDOWS\System32\WindowsPowerShell\v1.0\;
C:\WINDOWS\System32\OpenSSH\;C:\xampp\php;C:\Users\mlath\AppData\Local
\Microsoft\WindowsApps;C:\VSCode\Microsoft VS Code\bin
```

Backtick Operator

PHP supports one execution operator: backticks (``). (they are not single-quotes!) PHP will attempt to execute the contents of the backticks as a shell command; the output will be returned. Use of the backtick operator is identical to shell_exec().

Example

Take a look at the following example −

```
<?php
   $output = `dir *.php`;
   echo "<pre>$output</pre>";
?>
```

It will produce the following **output** −

```
Volume in drive C has no label.
Volume Serial Number is 7EE4-E492

Directory of C:\xampp\htdocs

10/26/2023  08:42 PM                61 hello.php
10/12/2023  10:40 AM                61 homepage.php
07/16/2015  09:02 PM               260 index.php
10/12/2023  10:39 AM                49 menu.php
09/25/2023  01:43 PM               338 myform.php
10/12/2023  10:49 AM                51 myname.php
10/26/2023  02:00 PM               369 test.php
09/25/2023  01:42 PM               555 uploadfile.php
               8 File(s)          1,744 bytes
               0 Dir(s)   168,471,289,856 bytes free
```

The backtick operator is disabled when **shell_exec()** is disabled.