**ლექცია № 13. ოპერაციები ფაილებზე**

როცა HTML-ფორმებით სერვერზე მიღებულია კლიენტის მხრიდან ინფორმაცია, რიგ შემთხვევაში საჭიროა ის ჩაიწეროს მეხსიერებაში და მერე მასზე გარკვეული ტიპის მანი­პულაციები შესრულდეს. რა თქმა უნდა საბოლოო შედეგია მონაცემთა ბაზებში ინფორ­მაციის ჩაწერა, მაგრამ პირველ ეტაპზე მოვახდინოთ ამ ინფორმაციის ჩაწერა ფაილში, რო­მელსაც ზოგჯერ ”ბრტყელ” - ორგანზომილებიან ფაილს უწოდებენ - ჩვენს შემთხვევაში გვექნება ტექსტური ფაილი. ყოველი კლიენტის შეკვეთა ჩაიწერება ფაილში ცალკე სტრი­ქონში.

დავიწყოთ შეკვეთებისთვის გამიზნული HTML-ფორმებით, რომელიც შემდეგი HTML კოდით მოიცემა -

ლისტინგი 1 შეკვეთის ფორმა index.html

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />

<title>ავტოსათადარიგო ნაწილები ბუბასგან</title>

</head>

<body>

<form action="processorder.php" method="post">

<table border="0">

<tr bgcolor="#cccccc">

<td width="140">საქონელი</td>

<td width="60">რაოდენობა</td>

</tr>

<tr>

<td>საბურავები</td>

<td align="left"><input type="text" name="tireqty" size="3"

maxlength="3" /></td>

</tr>

<tr>

<td>ზეთი</td>

<td align="left"><input type="text" name="oilqty" size="3"

maxlength="3" /></td>

</tr>

<tr>

<td>აალების სანთლები</td>

<td align="left"><input type="text" name="sparkqty" size="3"

maxlength="3" /></td>

</tr>

<tr>

<td>მიწოდების მისამართი</td>

<td align="left"><input type="text" name="address" size="50"

maxlength="50" /></td>

</tr>

<tr>

<td colspan="2" align="center"><input type="submit"

value="შეკვეთის გაგზავნა" /></td>

</tr>

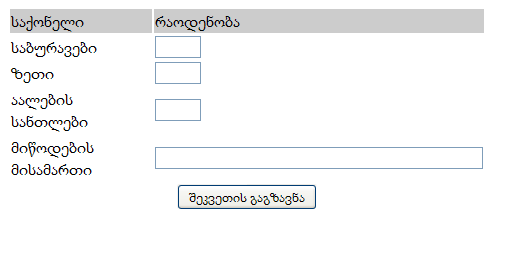
</table>

</form>

</body>

</html>

ლისტინგ 1-ის გაშვების შემდეგ მივიღებთ:



ნახატი 1 შეკვეთის ფორმა

როგორც ვხედავთ, ადრე არსებულ ფორმებს დაემატა ”მიწოდების მისამართი”-ს ფორმა, რომლის სახელია address (იხ. ლისტინგ 1-ის 30-ე სტრიქონში) და იგი შეკვეთის გაგზავნის შემდეგ ჩაიწერება ცვლადში რომლის იდენტიფიკატორებია: $\_REQUEST['address'] და $\_POST['address'] – ეს უკანასკნელი და არა $\_GET['address'] განპირობებულია იმით, რომ ფორმის ცვლადების გაგზავნად ლისტინგ 1-ში ( სტრიქონი 7) მითითებულია POST და არა GET მეთოდი.

დავისახოთ მიზნად, რომ ყოველი შეკვეთა ჩაიწეროს ერთსა და იმავე ფაილში, და მათ დასათვალიებრად შევქმნათ საჭირო ინტერფეისი.

იმისათვის, რომ მონაცემები ჩაიწეროს ფაილში საჭიროა:

1. გავხსნათ ფაილი. მაგრამ თუ კი ეს ფაილი არ არსებობს - შევქმნათ იგი.
2. ჩავწეროთ მონაცემები ფაილში.
3. დავხუროთ ფაილი.

ანალოგიურად, იმისთვის, რომ მონაცემები იქნეს წაკითხული ფაილიდან შესასრულებე­ლი ასევე სამი მოქმედებაა:

1. გავხსნათ ფაილი. მაგრამ თუ კი ამ ფაილის გახსნა შეუძლებელია (ვთქვათ ის არ არსებობს) საჭიროა ამ სიტუაციის გათვითცნობიერებაა და მდგომარეობიდან კორექტული გამოსავალის გასათვალისწინება.
2. წავიკითხოთ მონაცემები ფაილიდან.
3. დავხუროთ ფაილი.

**ფაილის გახსნა**

PHP-ში ცარიელი ფაილის შექმნისათვის გამოიყენება ფუნქცია touch(), რომლის ფრჩხილებში ჩაისმება ფაილის სახელი. მაგალითად, შევქმნათ ცარიელი ფაილი იმავე კატალოგში

ლისტინგი 2. ცარიელი ფაილის შექმნა

<!DOCTYPE html>

<html>

<body>

<?php

// ცარიელი ფაილის სახელი

$filename = "text.txt";

// ვქმნით ცარიელ ფაილს სახელით - $filename

touch($filename);

if (touch($filename)) echo 'ცარიელი ფაილი სახელით $filename წარმატებით შეიქმნა'

else echo 'ცარიელი ფაილი სახელით $filename ვერ შეიქმნა';

?>

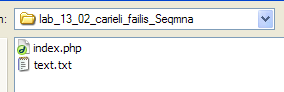
</body>

</html>

ლისტინგ 2-ის გაშვების შემდეგ ვღებულობთ:

ცარიელი ფაილი სახელით $filename წარმატებით შეიქმნა

მართლაც, კატალოგში lab\_13\_02\_carieli\_failis\_Seqmna შეიქმნა ფაილი text.txt.



ნახატი 2 ფაილი text.txt შეიქმნა

მაგრამ უმეტეს შემთხვევაში ფაილის შექმნა შერწყმულია ფაილის გახსნის ფუნქციასთან, როდესაც ფაილის სახელი დაკავშირებულია დისკრიპტორთან (მთელ რიცხვთან - უნიკალურს სკრიპტის ფარგლებში), რომელიც გამოიყენება ამ ფაილის წვდომისას. ფაილის სახელის დაკავშირებას დისკრიპტორთან ეწოდება ფაილის ”გახსნა” და ხდება fopen() ფუნქციით. მისი სინტაქსისია:

***resource* fopen($filename, $mode [, $use\_include\_path [, $context]])**

fopen() ფუნქცია გახსნის ფაილს სახელწოდებით $filename სათანადო - $mode რეჟიმში. ზოგიერთი რეჟიმი ითხოვს, რომ ეს $filename ფაილი უკვე უნდა იყოს შექმნილი. არააუცილე­ბელი პარამეტრი $use\_include\_path იძლევა საშუალებას მოცემულ იქნას გზა $filename ფა­ილის მოსაძებნად. პარამეტრი $context იძლევა საშუალებას ქსელურ ფაილთან ურთიერთო­ბისას სერვერმა გააგზავნოს HTTP- სათაურები. fopen() ფუნქციის წარმატებული შესრულები­სა ბრუნდება გახსნილი ფაილის დესკრიპტორი, წინააღმდეგ შემთხვევაში FALSE . fopen() მიამაგრებს სახელობით რესურს, რომელიც მითითებულია ჩაწერილი *filenam -ში,* ნაკადთან. PHP-ში ნაკადი (stream) — ეს რესურსია (resource), რომელიც ახდენა მონაცემთა უწყვეტი მიმდევრობის წყაროს იმიტირებას. ამდენად ნაკადიდან შესაძლებელია მონაცემთა წაკითხვა, და ასევა ჩაწერა. fopen(), თუ წარმატებით შესრულდა, მაშინ ის მიიღებს შემდეგი სახის მნიშვნელობას - მაგ. Resource id #3, და ის იქნება შემდეგ გამოყენებული გახსნილ ფაილზე მოქმედებების შესასრულებლად!

როგორც ვხედავთ, fopen() ფუნქციის პირველი არგუმენტია ფაილის სახელი $*filename -* ის ფაილი რომელიც უნდა გაიხსნას, ან ჯერ შეიქმნას და მერე ამავდრო­ულად გაიხსნას, რაზეც ოდნავ მოგვიანებით ვისაუბრებთ. შევეხოთ დაწვრილებით ფაილის სახელს. ერთი და იგივე სახელების ფაილები შეიძლება იყოს ჰოსტის მეხსიერების სხვადასხვა ნაწილში. ამიტომაც, როდესაც მიუთითებებ ფაილის სახელს, უნდა იქნეს ცხადად განსაზღვრული ჰოსტის თუ ინტერნეტ-ქსელური რესურსის ის ნაწილიც, რომელშიც არის განთავსებული კონკრეტული დასახელე­ბის ფაილი, რომელიც უნდა გაიხსნას. გავიხსენოთ ჰოსტში ფაილების განლაგების იერარქია და სტრუქტურა.

ფაილი წარმოადგენს მეხსიერებაში ჩაწერილ ბაიტების მიმდევრობას, რომელსაც თავისი სახელი - ლოკალური სახელი გააჩნია.

ფაილების ერთობლიობა შეიძლება გაერთიანებული იყოს კატალოგში - კატალოგიც მისი შემცველი ფაილის მეხსიერების ერთეულში იმოფება. აღსანიშნავია, რომ კატალოგი - ფოლ­დერი შეიძლება შეიცავდეს სხვა კატალოგს.

კატალოგთა ან ფაილთა (არცერთ კატალოგში არშემავალი ფაილი) ერთობლიობა მო­თავსებულია ტომში - ვთქვათ C,D,F და ა.შ. ტომთა ერთობლიობა ქმნის ჰოსტის მთლიან მეხსიერებას (ოპერატიულის გარდა). ამდენათ კატალოგებისა და ფაილების განლაგება ტომში არის „მატრიოშკას“ პრინციპით.

ვთქვათ სკრიპტი, რომელმაც უნდა გახსნას რომელიმე ფაილისახელით scr.php არის გან­თავსებული C ტომის კატალოგში - მისი **აბსოლუტური მისამართი** ფაილურ სისტემაში ჩა­იწერება შემდეგნაირად:

C:/ WWW/ scr.php (ნახაზ 1-ზე ღია ცისფერი)

ეს აბსოლუტური მისამართი ამ კონკრეტული ფაილისთვის არის ერთადერთი. თუ ზუს­ტად ამავე სახელით - scr.php არის კიდევ სხვა ფაილი, მაშინ მისი აბსოლუტური მისამართი იქნება სხვა.

თუ კი ეხლა სკრიპტმა scr.php უნდა გახსანას ფაილი test.txt , რომელიც მოთავსებულია C ტომის dir კატალოგში, საჭიროა სკრიპტში ჩაიწეროს მისი აბსოლურური მისამართი (ნახაზ 1-ზე ვარდისფერი ):

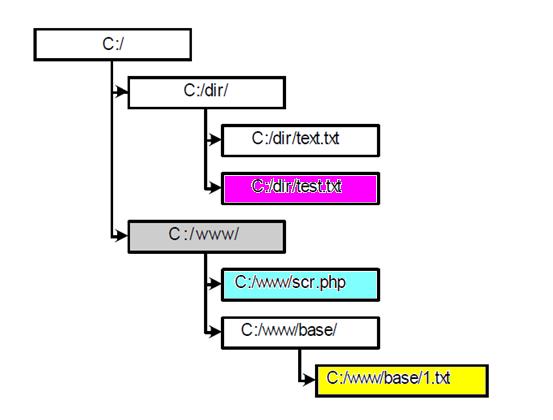
C:/ dir/ test.txt

აბსოლურური მისამართი არაა ერთადერთი გზა ფაილურ სისტემაში კონკრეტული ფა­ილის მოძებნისთვის. ზოგჯერ უფრო ეფექტურია ე.წ. ფარდობითი გზის გამოყენება. ფარდო­ბითობა აიღება იმ კატალოგის მიმართ რომელშიც მოთავსებულია სკრიპტი, ჩვენს შემთხვე­ვაში ეს კატალოგია C:/ WWW (ნახაზ 1-ზე ნაცრისფერი), რომელშიც მოთავსებულია სკრიპტი scr.php. განვიხილოთ ორი შემთხვევა:

ა) გასახსნელი ფაილი იმ კატალოგშია ამ მის ქვემდებარე კატალოგებში, რომელშიც იმყო­ფება სკრიპტი. მაშინ გზა ამ ფაილიდან ფარდობითი კატალოგიდან ჩაიწერება იმდენი სლე­შის გამოყენებით, თუ რამდენი დონით უნდა ჩავიდეთ ქვემოთ ამ ფარდობითი კატალოგი­დან. მაგალითად გზა ფარდობითი კადალოგიდან ფაილამდე 1.txt რომელიც მოთავსებულია უშუალოდ WWW კატალოგის base ქვეკატალოგში, ჩაიწერება:

base/1.txt

ჯერ დაიწერა კატალოგი base წინ სლეშის გარეშე, რადგანაც იგი scr.php სკრიპტის დონე­ზეა - ანუ იმავე WWW კატალოგშია. შემდეგ სლეში, რადგანაც base კატალოგში ”ჩავდი­ვართ” და ამის შემდეგ ფაილის სახელი 1.txt თავის გაფართოებით.

* 
* ნახაზი 3. ფაილების იერარქია

ბ)გასახსნელი ფაილი იმ კატალოგშია, რომელიც ფარდობითი კატალოგიდან წვეროსკენ მიმავალ აღმავალი ჯაჭვშია, ანაც სხვა ფაილური სისტემის სხვა განტოტებაში. ვთქვათ scr.php-დან უნდა გავხსნათ test.txt . მასში მისაღწევად WWW კატალოგიდან ჯერ უნდა გადავიდეთ მაღლა C ტომში, რისთვისაც ვიყენებთ .. წერტილს, ხოლო შემდეგ მივყვებით ა)-ში მითითებული სქემით /dir/test.txt. საბოლოოდ ვღებულობთ სრულ ფარდობით გზას:

.. /dir/test.txt.

რამდენაცჯერ უნდა ”ავიდეთ მაღლა” იმდენჯერვე უნდა გამოვიყენოთ ../ .

WEB-ში ფაილური სისტემების გზის გარდა გამოიყენება ქსელური გზაც, რისთვისაც საჭიროა პროტოკოლის და URL-ის მითითება. მაგალითად <http://www.softtime.ru/index.php>

ამ შემთხვევაში ინტერპრეტატორი ეძებს მითითებული პროტოკოლის დამმუშავებელს. თუ კი დამმუშავებელი ნაპოვნია, ამის შემდეგ PHP ამოწმებს - დაშვებულია თუ არა URL ობიექტებზე ისეთივე მოქმედებები, როგორც ჰოსტის ჩვეულებრივ ფაილებთან - ეს დაშვება არის დირექტივა - allow\_url\_fopen. თუ კი მითითებულია allow\_url\_fopen=off მაშინ fopen() ფუნქცია გამოიწვევს შეცდომას და მივიღებთ გაფრთხილებას. თუ კი ფაილის გზა ატ იწყება პროტოკოლით, ინტერპრეტატორი ჩათვლის, რომ საქმე გვაქვს ჰოსტის ლოკალურ ფაილთან და ხდება იმის დადგენა, არის თუ არა ამ ფაილის გახსნის უფლება.

fopen() -ის პარამეტრი use\_include\_path მნიშვნელობის 1 ან TRUE აიძულებს ინტერპრეტა­ტორს ეძებოს fopen()-ში მითითებული ფაილი include\_path -ში. include\_path გახლავთ დირექტივა PHP-ს მომართვის ფაილიდან, რომელიც მიუთითებს კატალოგების იმ სიას, რომლებშიც შეიძლება იყოს ჩართვისთვის საჭირო ფაილები. ეს პროცედურა გამოიყენება include() და require() ფუნქციებისთვისაც.

ცხრილი 1. fopen() ფუნქციის მოქმედებები რეჟიმების გათვალისწინებით

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| რეჟიმი | წაკითხვა | ჩაწერა | ფაილური მიმთითებელი | ფაილის გასუფთავება | შევქმნათ, თუ კი ფაილი არ არის | შეცდომა, თუ კი ფაილი არის |
| r | დიახ | არა | თავში | არა | არა | არა |
| r + | დიახ | დიახ | თავში | არა | არა | არა |
| w | არა | დიახ | თავში | დიახ | დიახ | არა |
| w+ | დიახ | დიახ | თავში | დიახ | დიახ | არა |
| a | არა | დიახ | ბოლოში | არა | დიახ | არა |
| a+ | დიახ | დიახ | ბოლოში | არა | დიახ | არა |
| x | არა | დიახ | თავში | არა | დიახ | დიახ |
| x+ | დიახ | დიახ | თავში | არა | დიახ | დიახ |
|  |  |  |  |  |  |  |

ჩავწეროთ ფორმის ცვლადები უკვე შექმნილ ფაილში text.txt . ჩვენ ამისთვის ვქმნით ფოლდერს lab\_13\_03\_Cawera\_failSi , რომელშიც არ არის ფაილი text.txt, იგი იმყოფება ფოლდერში lab\_13\_02\_carieli\_failis\_Seqmna. lab\_13\_03\_Cawera\_failSi და lab\_13\_02\_carieli\_failis\_Seqmna კატალოგები ერთი დონისაა და მოთავსებულია კატალოგში: laboratoriuli\_13.

ამდენად lab\_13\_02\_carieli\_failis\_Seqmna-ში ჩავწეროთ შეკვეთის ფორმა ლისტინგ 1-ით. ამავე კატალოგშია სკრიპტი processorder.php, რომელიც დაამუშავებს ფორმის ცვლადებს და ჩაწერს text.txt ფაილში (სხვა კატალოგში) შეკვეთას.

ლისტინგი 3. სხვა კატალოგში მოთავსებულ ცარიელ ფაილში ჩაწერა

<html>

<head>

<title>ავტოსათადარიგო ნაწილები ბუბასგან - შეკვეთის შედეგი</title>

</head>

<body>

<h1>ავტოსათადარიგო ნაწილები ბუბასგან</h1>

<h2>შეკვეთის შედეგი</h2>

<?php

// ცვლადების მოკლე სახელების შექმნა

$tireqty = $\_POST['tireqty'];

$oilqty = $\_POST['oilqty'];

$sparkqty = $\_POST['sparkqty'];

$address = $\_POST['address'];

$DOCUMENT\_ROOT = $\_SERVER['DOCUMENT\_ROOT'];

$date = date('H:i, jS F Y');echo "<p>შეკვეთა დამუშავებულია.$date.</p>";

$totalqty = $tireqty + $oilqty + $sparkqty;

echo "შეკვეთილია საქონელი ".$totalqty."<br />";

if ($totalqty == 0) {

echo "თქვენ არაფერი არ შეგიკვეთიათ წინა გვერდზე<br />";

} else {

if ($tireqty > 0) {

echo $tireqty." საბურავი<br />";

}

if ($oilqty > 0) {

echo $oilqty." ბოთლი ზეთი<br />";

}

if ($sparkqty > 0) {

echo $sparkqty." აალების სანთელი<br />";

}

}

define('TIREPRICE', 100);

define('OILPRICE', 10);

define('SPARKPRICE', 4);

$totalamount = $tireqty \* TIREPRICE

+ $oilqty \* OILPRICE

+ $sparkqty \* SPARKPRICE;

$totalamount = number\_format($totalamount, 2, '.', ' ');

echo "<p> ჯამში შეკვეთით: $".$totalamount." </p>";

echo "<p> შეკვეთის მისამართი: ".$address." </p>";

$outputstring = $date . "\t" . $tireqty . " საბურავი\t"

. $oilqty. " ბოთლი ზეთი\t" . $sparkqty . " აალების სანთელი\t\$"

. $totalamount . "\t" . $address . "\n";

// ვხსნით ფაილს ჩაწერის დასამატებლად

$fp = fopen**("../lab\_13\_02\_carieli\_failis\_Seqmna/text.txt", "w");**

if (!$fp) {

echo "<p> <strong> ამ მომენტში ჩვენ არ შეგვიძლი თქვენი შეკვეთის დაკმაყოფილება. "

. "გთხოვთ გაიმეოროთ შეკვეთა მოგვიანებით. </strong> </p> </body></html>";

exit;

}

flock($fp, LOCK\_EX);

fwrite($fp, $outputstring, strlen($outputstring));

flock($fp, LOCK\_UN);

fclose($fp);

echo "<p> შეკვეთა ჩაწერილია.</p>";

?>

</body>

</html>

52-ე სტრიქონში, წითლად გამოყოფილში, ჩაწერილია ფაილის გახსნის და ჩაწერის ფუნ­ქცია რეჟიმით “w”. ამის შემდეგ ჩვენ უნდა გახსნილ ფაილზე რაიმე მოქმედება შევასრულოთ, კერძოდ მასში ჩავწეროთ რაიმე მნიშვნელობა. ამისთვის გამოიყენა ფუნქცია fwrite($fp, $outputstring, strlen($outputstring)); ზოგადად მას აქვს შემდეგი სინტაქსისი:

fwrite($handle, $str [, $length])

fwrite() ფუნქცია გახსნილ ფაილში რომელსაც $handle მიმთითებელი აქვს, ჩაწერს $str სტრიქონის მნიშვნელობას. არააუცილებელი პარამეტრი $length იძლევა საშუალებას განსაზ­ღვრულ იქნეს ბაიტების ის რაოდენობა, რომელიც უნდა ჩაიწეროს - ამ რაოდენობის მიღწე­ვისას, ჩაწერა დასრულდება, ანა ჩაწერა დასრულდება მაშინ, როცა ფაილის ბოლოში მოხდება გასვლა. წარმატების შემთხვევაში fwrite() ფუნქცია აბრუნებს ჩაწერილი ბაიტების რაოდენო­ბას, წარუმატებლობის შემთხვევაში FALSE -ს. fwrite() ფუნქციას აქვს ფსევდონიმი - fputs()- ეს ფუნქციაც ისევე მოქმედებს, როგორც fwrite().

ჩვენს შემთხვევაში $fp მიმთითებლით განსაზღვრულ გახსნილ ფაილში ჩაიწერება $outputstring სტრინგის მნიშვნელობას, ხოლო ჩაწერილი ბაიტების რაოდენობა იქნება $outputstring სტრინგის სიგრძის ტოლი - რომელიც strlen($outputstring) ფუნქციით განისაზღვრება.

ფუნქცია fwrite()-ია ანალოგია **file\_put\_contents()**, რომელსაც საერთოდ არ სჭირდება წინასწარ ფაილის გახსნა და დაკეტვა, და არც დისკრიპტორიც ჩასმა, ის ფაილის სახელს ითხპვს და იმ მონაცემებს, რომელიც ინდა ჩაიწეროს. მისის სინტაქსისია:

**file\_put\_contents($filename, $data [, $flags [, $context]]);**

$filename - ფაილის სახელია, რომელშიც უნდა მოხდეს ჩაწერა, $data - ის რაც უნდა ჩაიწეროს. არააუცილებელი პარამეტრები $flags და $context გამოიყენება დაშორებულ ფაილებში ჩაწერისას, მაგალითად, როცა საჭიროა HTTP და FTP პროტოკოლებისგამოძახება.

გახსნილ ფაილის მიმართ დასახული ოპერაციების შესრულებისასს, მაგალითად ფაილის წაკითხვა; ფაილში ჩაწერა ან სხვა, მაშინ შეიძლება ეს ფაილი დაიხუროს, რისთვისაც გამოიყენება ფუნქცია fclose (), რომლის სინტაქსისია:

**fclose ($fp)**

სადაც $fp - გახსნილი ფაილის მიმთითებელია.

ამდენად ჩვენს შემთხვევაში, ლისტინგი 3-ის შესრულებისას გაიხსნება ფაილი text.txt, შემდეგ ფაილში text.txt, რომელიც ადრე ცარიელი იყო, ჩაიწერება:

**16:17, 15th December 2014 5 საბურავი 8 ბოთლი ზეთი 6 აალების სანთელი $604.00 Sokhumi Nazadze 33, 7**

ამის შემდეგ ფაილი text.txt დაიხურება

“w” რეჟიმი ფაილში ჩაწერის წინ ამ ფაილს ასუფთავებს. ამიტომ შემდეგში, როცა იგივე ფაილში საჭიროა არსებულ ჩანაწერს ახალი დავუმატოთ, ამ რეჟიმს ვერ გამოვიყენებთ, ამ მიზნისათვის საჭიროა გამოვიყენოთ რეჟიმი “a” ან “a+”.

ამისთვის ლისტინგ 3-ში, ფაილ processorder.php -ის 52-ე სტრიქონში უნდა ჩავწეროთ:

$fp = fopen**("../lab\_13\_02\_carieli\_failis\_Seqmna/text.txt", "a");**

ამ ამოცანის რეალიზაცია მოვახდი­ნოთ ფოლდერში lab\_13\_04\_failSi\_Canaweris\_damateba. შედეგად text.txt ფაილში ძველ ჩანაწერს ახალი დაემატება (ის წითლადაა გამოყოფილი):

**16:17, 15th December 2014 5 საბურავი 8 ბოთლი ზეთი 6 აალების სანთელი $604.00 Sokhumi Nazadze 33, 7**

**16:37, 15th December 2014 12 საბურავი 18 ბოთლი ზეთი 24 აალების სანთელი $1 476.00 Tbilisi, Politkovskaia 12**

იმისათვის რომ fopen() ფუნქციამ შეცდომა არ გამოიტანოს მონიტორზე მისი მიჭირხვნისთვის გამოიყენება სიმბოლო @ -და შეგვეძლო ჩაგვეწერა

@ $fp = fopen("../lab\_13\_02\_carieli\_failis\_Seqmna/text.txt", "a");

მოდით ეხლა წარმოვიდგინოთ, რომ ინტერნეტ შეკვეთას აკეთებს რამდენიმე კლიენტი ერთდროულად და ასევე ერთდროულად ხდება ამ შეკვეთების დამუშავება. ანუ ხდება ერთი და იმავე ფაილზე მიმართვა - მისი გახსნა, მასში ჩაწერა და ეს ყველაფერი ხდება დროის ერთ მომენტში. ამ შემთხვევაში დიდია ალბათობა, რომ მოხდეს შეცდომა და შეკვეთების არას­წორი ჩანაწერები მივიღოთ. ამ სიტუაციიდან თავის დასაღწევად გამოიყენება ფუნქცია, რო­მელიც ბლოკავს ამ ფაილის გამოყენებას სხვა რომელიმე დამუშავებისათვის, მანამ მიმდი­ნარე დამუშავება არ დასრულდება. ამის შემდეგ შესაძლებელია მოიხსნას ბლოკი ამ ფაილი­დან. ბლოკირების ფუნქციას აქვს შემდეგი სინტაქსისი:

*bool* flock(*resource* fp, *int* operation[, *int* &wouldblock]) ;

flock() ფუნქციას გადაეცემა გახსნილი ფაილის მიმთითებელი *resource* fp და კონსტანტა *int* operation რომელიც განსაზღვრავს საჭირო ბლოკირების ტიპს. flock() ფუნქცია აბრუნებს true, თუ ბლოკირება წარმატებით სესრულდა და false-ს წინააღმდეგ შემთხვევაში. არააუცილებელი პარამეტრი *int* &wouldblock უნდა შეიცავდეს true-ს, თუ კი მოთხოვნილ ბლოკირებას შეუძლია მიმდინარე პროცესის ბლოკირება.

Ooperation პარამეტრების მნიშვნელობები PHP-4.0.1 ვერსიაში შეიცვალა, ამიტომ ეს ცვლილებები მოტანილია ცხრილ 2-ში

ცხრილი 2. flock() ფუნქციის operation პარამეტრების მნიშვნელობები

|  |  |
| --- | --- |
| **operation პარამეტრების მნიშვნელობები** | **აღწერა** |
| LOCK\_SH  (ადრინდელ ვერსიებში 1) | წაკითხვის ბლოკირება. ფაილი შეიძლება გამოყენებულ იქნეს ერთდროულად სხვადასხვა წამკითხავი აპლიკაციების მიერ |
| LOCK\_EX  (ადრინდელ ვერსიებში 2) | ჩაწერის ბლოკირება. ეს მონოპოლური რეჟიმია. ფაილი ერთობლივი სარგებლობისთვის მიუწვდომელია |
| LOCK\_UN  (ადრინდელ ვერსიებში 3) | არსებული ბლოკირების გაუქმება |
| LOCK\_NB  (ადრინდელ ვერსიებში 4) | მიმდინარე ბლოკირებისას ხდება სხვა, დამატებითი ბლოკირების მცდელობის თავიდან აცილება |

ცხადია ბლოკირება და ბლოკირებისგან მოხსნა უნდა მოხდეს ფაილის გახსნის შემდეგ, და მათ შორის უნდა იყოს რაიმე მანიპულაცია გახსნილ ფაილში, ისევე, როგორც ეს არის ლისტინგ 3-ში:

$fp = fopen**("../lab\_13\_02\_carieli\_failis\_Seqmna/text.txt", "w");**

if (!$fp) {

echo "<p> <strong> ამ მომენტში ჩვენ არ შეგვიძლი თქვენი შეკვეთის დაკმაყოფილება. "

. "გთხოვთ გაიმეოროთ შეკვეთა მოგვიანებით. </strong> </p> </body></html>";

exit;

}

flock($fp, LOCK\_EX);

fwrite($fp, $outputstring, strlen($outputstring));

flock($fp, LOCK\_UN);

შევნიშნოთ, რომ flock() ფუნქცია არ მუშაობს სხვადასხვა ქსელურ ფაილურ სისტემებში, მათ შორის - NFS (Nerwork File System). ის ასევე არ მუაობს FAT (File Allocation Table - ფაილების განლაგების ცხრილი)

**წაკითხვა ფაილიდან**

როცა მომხმარებელი ინტერნეტის საშუალებით გააკეთებს შეკვეთებს, ისინი ჩაიწერება სპეციალურ ფაილში და იმისთვის, რომ ეს შეკვეთები ნახოს ავტონაწილების მაღაზიის თანამშრომელმა, ეს ფაილი სპეციალური მეთოდებით, დამოუკიდებლად უნდა გახსნას. მაგრამ შესაძლებელია, რომ შეიქმნას ვებ-ინტერფეისი, რომელიც მოგვცემს საშუალებას ადვილად იქნეს წაკითხული შეკვეთების ეს ფაილი.

ლისტინგი 4. შეკვეთების ფაილის წაკითხვა

<html>

<head>

<title>ავტოსათადარიგო ნაწილები ბუბასგან - კლიენტების შეკვეთები</title>

</head>

<body>

<h1>ავტოსათადარიგო ნაწილები ბუბასგან</h1>

<h2>კლიენტების შეკვეთები</h2>

<?php

//ცვლადების მოკლე სახელების შექმნა

$DOCUMENT\_ROOT = $\_SERVER['DOCUMENT\_ROOT'];

@ $fp = fopen("orders.txt", 'rb');

if (!$fp) {

echo "<p><strong>არაა დაუმუშავებელი შეკვეთები.

შემოიარეთ მოგვიანებით.</strong></p>";

exit;

}

while (!feof($fp)) {

$order= fgets($fp, 999);

echo $order."<br />";

}

echo 'საბოლოო პოზიცია მითითებულ ფაილში: '.(ftell($fp));

echo '<br />';

rewind($fp);

echo 'პოზიცია, მას მერე რაც გამოძახებულია ფუნქცია rewind(): '.(ftell($fp));

echo '<br />';

fclose($fp);

?>

</body>

</html>

შედეგად მივიღებთ

# ავტოსათადარიგო ნაწილები ბუბასგან

## კლიენტების შეკვეთები

00:28, 17th December 2014 12 საბურავი 8 ბოთლი ზეთი 6 აალების სანთელი $1 304.00 სოხუმი, ნოზაძის 33, ბ. 7   
00:30, 17th December 2014 8 საბურავი 6 ბოთლი ზეთი 12 აალების სანთელი $908.00 ქუთაისი, დ. აღმაშენებლის 4, ბ. 18   
00:30, 17th December 2014 4 საბურავი 12 ბოთლი ზეთი 6 აალების სანთელი $544.00 თბილისი, პოლიტკოვსკაიას 12   
  
საბოლოო პოზიცია მითითებულ ფაილში: 618  
პოზიცია, მას მერე რაც გამოძახებულია ფუნქცია rewind(): 0

ეხლა განვიხილოთ დაწვრილებით ის ფუნქციები, რომელიც ლისტინგ4-ში

ფუნქცია fopen()-ით ჩვენ კვლავ ვახდენთ ფაილის გახსნას, რომელიც ამჟამად ჩაწერილია ფოლდერში lab\_13\_06\_SekveTis wakiTxva და მისი სახელია "orders.txt". სწორედ ამ ფაილში მოხდა სამი შეკვეთის ჩაწერა და ამჟამად ლისტინგ 4-ში ხდება ამ ჩანაწერების წაკითხვა სკრიპტში - vieworders.php.

ამისთვის ფუნქცია fopen()-ში რეჟიმი მითითებულია ”r” - წაკითხვა, ”b”, რომელიც ”r” -ის გვერდითაა მიუთითებს არის ბინალური რეჟიმი, რომელიც ერთ ერთ ძირითად რეჟიმთან - გამოიყენება მაშინ, როცა ფაილური სისტემა განასხვავებს ბინარული და ტექსტურ ფაილებს. Windows ოპერაციული სისიტემა განასხვავებს ამ ფაილებს, UNIX - არა. PHP-ის შემქმნელები იძლევიან რეკომენდაციას ”b”-ს გამოყენებისა, როგორც მაქსიმალური გადატანისათვის. იგი ჩუმატობის პრინციპით გამოიყენება. ”b” -ს მაგივრად დამატებაში შეიძლება იყოს ”t” , რაც აღნიშნავს ტექსტურ რეჟიმს, Windows ოპერაციული სისიტემისთვის ის აქტუალურია.

ლისტინგ 4-ში გამოყენებულია ციკლი while იმისთვის, რომ წავიკითხოთ ფაილიდან, მანამ სანამ არ იქნება ფაილის ბოლო. სწორედ ციკლი while ამოწმებს, მიღწეულია თუ არა ფაილის ბოლო feof() ფუნქციის საშუალებით:

While(!feof($fp))

ფუნქცია feof() არგუმენტია ფაილის დისკრიპტორი, აბრუნებს true-ს, თუ ფაილის მიმთითებელმა მიაღწია ფაილის ბოლოს, წინააღმდეგ შემთხვევაში false-ს. ჩვენს შემთხვევაში სკრიპტ vieworders.php. ეს ციკლი გაგტძელდება მანამ, სანამ მიმთითებელი არ მიაღწევს სიმბოლოს **EOF** – (End Of File).

ფაილის წასაკითხად ლისტინგ 4-ში გამოიყენება ფუნქცია fgets():

$order= **fgets($fp,999);**

ეს ფუნქცია fgets() - გამოიყენება ყოველი მიმართვისთვის ფაილის ერთი სტრიქონის-ჩანაწერის წასაკითხად. კერძოდ წაკითხვა გაგრძელდება მანამ, სანამ არ შეგვხდება ახალი სტრიქონის სიმბოლო (\n), ან სიმბოლო EOF, ან ფაილიდან არ იქნება წაკითხული 998 ბაიტი - რაც წარმოადგენს წასაკიღხი სტრიქონის მაქსიმალურ სიგრძეს - მითითებულ მთელ რიცხვს გამოკლებული ერთი. ფუნქცია fgets() ძირითადად გამოიყენება ტექსტური ფაილების წასაკითხად, მაგრამ ასევე არსებობს მისი მსგავსი - ფაილის წამკითხავი ფუნქციები. მაგალითად:

fgetc () - ფუნქცია გამოიყენება ფაილის თითოეული სიმბოლოს წასაკითხად.

fread () - ფუნქცია, რომელიც კითხულობს ფაილიდან ნებისმიერი რაოდენობის ბაიტებს, სინტაქსისი: ***string* fread($myfile, filesize("order.txt"));**

fgetss( ) - ფუნქცია დააბრუნებს ერთ სტრიქონს HTML და PHP ტეგების გარეშე. სინტაქსი­სი: **fgetss(file,length,tags**) - თუ კი საჭიროა რომელიმე ტეგების დატოვება, ისინი მესამე პარა­მეტრში ჩაიწერება.

fgetcsv() - ფუნქცია ანაწევრებს ფაილის სტრიქონს რაიმე სიმბოლოს- სეპარატორის საშუ­ალებით, სინტაქსისია **fgetcsv(file,length,separator,enclosure**). Enclosure განსაზღვრავს, თუ რაში მოთავსდება გამოყოფილი ნაწილი,ჩუმატობის პრინციპით გვაქვს სინბოლო ბრჭყალები -” .

თუ კი ფაილის სიგრძე წინასწარ ცნობილია, ფაილის წაკითხვა შეიძლება ერთჯერადი მი­მართვით ფუნქცია fread()-ის საშუალებით. მისი სინტაქსისია:

**fread($handle, $length);**

$handle ფაილის დისკრიპტორია, $length - ფაილის სიგრძე. ფუნქცია fread() აბრუნებს სტრინგს, რომელშიც ფაილიდან წაკითხული ინფორმაცია. წაკითხვა კი განისაზღვრება მითითებული სიგრძით ან მანამ ფაილის ფოლოში არ იქმნება გასვლა. ამ ფუნქციისთვის წასაკითხი ბაიტების რაოდენობა აუცილებლად უნდა იქნეს მითითებული, ამდენად თუ კი გვსურს fread() ფუნქციით იქნეს წაკითხული მთლიანი ფაილი, საჭიროა ამ ფაილის სიგრძის ცოდნა, რაშიც დაგვეხმარება ფუნქცია filesize($filename)- რომელიც მხოლოდ ლოკალურ ფაილებზე მუშაობს. filesize() ფუნქცია ახდენს თავისი მუშაობის შედეგის კეშირებას. ანუ თუ მოხდა ფაილის შეცვლა, მაშინ filesize() ფუნქცია იგივე შედს მოგვცემს, ეს ტომ არ მოხდეს საჭიროა სტატიკური კეშის გასუფთავება clearstatcache() ფუნქციის საშუალებით.

ფაილის მთლიანი წაკითხვისას არაა საჭირო მისი დისკრიპტორის მითითება თუ გამოვიყე­ნებთ ფუნქცია file\_get\_contents() -ს , რომელსაც აქვს სინტაქსისი:

**file\_get\_contents($filename [, $flags [, $context [, $offset [, $maxlen]]]])**

$flags - მეორე პარამეტრი მიუთითებს რეჟიმს

ცხრილი 3 file\_get\_contents() ფუნქციის $flags პარამეტრის მნიშვნელობები

|  |  |
| --- | --- |
| **მნიშვნელობა** | **აღწერა** |
| FILE\_USE\_INCLUDE\_PATH | კონფიგურაციული ფაილის php.ini -ს include\_path დირექტივში მითითებულ კატალოგებში ფაილის ძებნა |
| FILE\_TEXT | მონაცემები ბრუნდება UTF-8 კოდირებაში (თუ კი ეს რეჟიმი ჩართულია კონფიგურაციული ფაილში php.ini) |
| FILE\_BINARY | წაკითხვა ხდება ბინარულ რეჟიმში, ფაილის შიგთავსის გარდაქმნის გარეშე |

გამოიყენება მცირე ზომის ფაილების წასაკითხად.

მთლიანი ფაილის წაკითხვა შესაძლებელია readfile() ფუნქციის საშუალებით, რომელიც თავად გახსნის ფაილს, წაიკითხავს მის შიგთავს, გამოიტანს ბრაუზერში და დახურავს მას. სინტაქსისია;

**readfile($filename [,$use\_include\_path [, $context]])**

readfile() ფუნქცია აბრუნებს წაკითხული ბაიტების რაოდენობას, შეცდომისას აბრუნებს FALSE -ს.

**ფაილების არსებობის შემოწმება**

იმისათვის, რომ ჩვენს მიერ აღწერილი ფუნქციების მოქმედება ფაილებზე განხორცი­ელდეს ეს ფაილები უნდა არსებობდნენ.თუ კი მითითებული ფაილები არ არსებობს PHP-ის ფუნქციები გამოიტანენ შეცდომას. ამდენად საჭიროა ისეთი ფუნქციების არ­სებობაც, რომლებიც ამ გამოტანილ შეცდომებს დაამუშავებს, ამისთთვის კი არსებობს ფუნქციები file\_exists(),   is\_writable(),   is\_readable() . (რომლებიც დაადგენენ, არსე­ბობს, თუ არა მითითებული ფაილი)

**ფუნქცია file\_exists()**

სინტაქსისი

*bool* file\_exists ($filename or $dirname)

ფუნქცია file\_exists() ამოწმებს, არსებობს თუ არა ის ფაილი ან დირექტორია (კატა­ლოგი), რომელიც გადაეცა მას, როგორც პარამეტრი. თუ მითითებული ფაილი ან დირექტორია ატსებობს, მაშინ ფუნქცია აბრუნებს TRUE-ს, წინააღმდეგ შემთხვევაში - FALSE-ს. ხდება ფუნქცია file\_exists()-ის მუშაობის კეშირება, ამიტომ სტსტიკური კე­შის გასასუფთავებლად, აქაც უნდა გამოვიყენოთ ფუნქცია clearstatcache(). არალოკალური ფაილებისთვის file\_exists() ფუნქცია არ გამოიყენება.

ლისტინგი 5 . ფაილის არსებობის შემოწმება

<!DOCTIPE HTML>

<HTML> <BODY>

<?php

$filename = 'x:/home/localhost/www/laboratoriuli\_13/lab\_13\_06\_SekveTis wakiTxva/orders.txt';

if (file\_exists($filename)) {

print "ფაილი <b>$filename</b> არსებობს";

} else {

print "ფაილი <b>$filename</b> არ არსებობს";

}

?>

</BODY></HTML>

ლისტინგი 5-ის გაშვებით მივიღებთ

ფაილი **x:/home/localhost/www/laboratoriuli\_13/lab\_13\_06\_SekveTis wakiTxva/orders.txt**არსებობს

**ფუნქცია is\_writable()**

თუ კი ფაილის ან დირექტორიის არსებობის გარდა დასადგენია, შეიძლება, თუ არა მასში ჩაწერაც, გამოიყენება ფუნქცია is\_writable() ან მისი სინონიმი is\_writeable(). სინტაქსისი:

*bool* is\_writable ($filename or $dirname)

ფუნქცია file\_exists() ამოწმებს, არსებობს თუ არა ის ფაილი ან დირექტორია(კატალოგი), რომელიც გადაეცა მას, როგორც პარამეტრი და ამასთან შესაძლებელია თუ არა მასში ჩაწერა. თუ მითითებული ფაილი ან დირექტორია არსებობს და მასში შეიძლება ჩაწერაც, მაშინ ფუნქცია აბრუნებს TRUE-ს, წინააღმდეგ შემთხვევაში - FALSE-ს. ფაილზე წვდომა მომხმარებლის იმ აღრიცხვის ჩანაწერებით, რომლითაც მუშაობს სერვერი(უფრო ხშირად ესაა nobody ან www). ხდება ფუნქცია file\_exists()-ის მუშაობის შედეგის კეშირება.

**ფუნქცია is\_readable()**

თუ კი ფაილის ან დირექტორიის არსებობის გარდა დასადგენია, შეიძლება, თუ არა მისი წაკითხვა, გამოიყენება ფუნქცია is\_readable(). სინტაქსისი:

*bool* is\_ readable ($filename or $dirname)

ეს ფუნქცია მუშაობს ფუნქცია is\_writable()-ს ანალოგიურად.

ლისტინგი 6. ფაილის არსებობისა და მისი წაკითხვის შესაძლებლობა

<!DOCTIPE HTML>

<HTML> <BODY>

<?php

$filename = 'x:/home/localhost/www/laboratoriuli\_13/lab\_13\_06\_SekveTis wakiTxva/orders.txt';

if (is\_readable($filename)) {

print "ფაილი <b>$filename</b> არსებობს

და წვდომადია წასაკითხად";

} else {

print "ფაილი <b>$filename</b>

ან არ არსებობს ან არაა წვდომადი წასაკითხად";

}

?>

</BODY></HTML>

ლისტინდ 6-ის გაშვებისას - <http://localhost/laboratoriuli_13/lab_13_08_failis_arseboba_da_wakiTxva> მივიღებთ

ფაილი **x:/home/localhost/www/laboratoriuli\_13/lab\_13\_06\_SekveTis wakiTxva/orders.txt** არსებობს და წვდომადია წასაკითხად

**ფაილის წაშლა**

იმისათვის, რომ PHP-ის საშუალებით მოვახერხოთ ფაილის წაშლა, უნდა გამოვიყენოთ ფუნქცია unlink(). სინტაქსისი:

*bool* unlink ($filename)

ლისტინგი 7. ფაილის წაშლა

<!DOCTIPE HTML>

<HTML> <BODY>

<?php

$filename = 'HTMLuri.html';

unlink($filename);

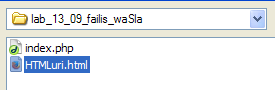
//ვშლით ფაილს სახელით

// 'HTMLuri.html';

?>

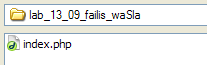
</BODY></HTML>

სანამ ლისტინგ 7-ს გავუშვებთ კატალოგში გვაქვს გაილები:



ნახატი 1. ფაილი წაშლამდე

ლისტინგ 7-ის გაშვებისას გამოყოფილი ფაილი წაიშლება და მივიღებთ:

****

ნახატი 2. ფაილი წაიშალა

**ფაილის ატვირთვა სერვერზე**

იმისათვის, რომ ავტვირთოთ ფაილი სერვერზე, საჭიროა შევქმნათ HTML ფორმა, და რათა ამ ფორმის მეშვეობით მოვახდინოთ ფაილის ატვირთვა სერვერზე, ფორმა უნდა შეიცავდეს ტეგ form-ის ატრიბუტ enctype-ს, რომლის მნიშვნელობაა multipart/form-data, ასევე file ტიპის input-ს .

ლისტინგი 8

<form enctype="multipart/form-data" action="parse.php" method="post">

<input type="hidden" name="MAX\_FILE\_SIZE" value="30000" />

აიტვირთება ფაილი: <input type="file" name="myfile" /><br>

<input type="submit" value="გააგზავნეთ ფაილი" />

</form>

დამატებულია ფარული ველი, რომელიც შეიცავს ასატვირთი ფაილის მაქსიმალურად და­საშვები ზომა ბაიტეში. თუ კი იქნება მცდელობა უფრო დიდი ზომის ფაილის ატვირთვისა წარმოიქმნება შეცდომა. ბრაუზერში შექმნილ ფორმას ექნება ტექსტის შეყვანის სახე, რომელ­საც აქვს დამატებითი ღილაკი ლოკალური დისკიდან ფაილის არჩევისთვის, სადაც ავირჩიეთ The Beatles.docx.



ნახატი 3. ასათვირთი ფაილის არჩევა

ამის შემდეგ შესაქმნელია სკრიპტი, რომელიც დაამუშავებს ატვირთულ ფაილს. მთელი ინფორმაცია ასატვირთი ფაილისა არის მოთავსებული გლობალურ მასივში $\_FILES. თუ ჩართულია დირექტივა *register\_globals* , მაშინ გადაგზავნილი ცვლადების მნიშვნელობები მათი სახელებითაა მიღწევადი. თუ კი კლიენტის კომპიუტერიდან ავტვირთავთ ფაილს სახელით *critics*.htm, რომლის ზომაა 15136 ბაიტი, მაშინ სკრიპტი ერთადერთი ბრძანებით print\_r($\_FILES) მონიტორზე გამოიტანს შემდეგს:

Array ( [myfile] => Array ( [name] => The Beatles.docx

[type] => application/vnd.openxmlformats-officedocument.wordprocessingml.document [tmp\_name] => X:\tmp\php8D.tmp

error] => 0

[size] => 16166

)

)

$\_FILES მასივს აქვს შემდეგი ელემენტები:

* $\_FILES['myfile']['name'] - სახელი, რომელიც ჰქონდა ფაილს კლიენტის ჰოსტზე.
* $\_FILES['myfile']['type'] - გაგზავნილი ფაილის mime-ტიპი, თუ კი ბრაუზერმა ეს ინფორმაცია მიაწოდა. ჩვენს შემთხვევაში ეს არის application/vnd.openxmlformats-officedocument.wordprocessingml.document.
* $\_FILES['myfile']['size'] - ჩარვირთული ფაილის ზომა ბაიტებში.
* $\_FILES['myfile']['tmp\_name'] - ფაილის დროებითი სახელი, რომლითაც ის შენახულია სერვერზე.
* $\_FILES['myfile']['error'] - შეცდომის კოდი, რომელიც ჩატვირთვის წარმოიქმნა.

 'myfile' - ფორმის ელემენტის სახელი, რომლის საშუალებითაც მოხდა ფაილის ჩატვირთვა სერვერზე. შეიძლება სხვა სახელიც მივანიჭოთ. მაგრამ სხვა გასაღებები ( name, type и т. д.) უცვლელი რჩება ყოველი ფორმისთვის.

თუ *register\_globals*=On, მაშინ შესაძლებელია სხვა ცვლადების წვდომაც, მაგალითად როგორიცაა $myfile\_name, რომელიც $\_FILES['myfile']['name']-ს ექვივალენტურის და ა.შ.

  *PHP* -ში ჩარვირთვის შეცდომა 5 ტიპისაა, შესაბამისად $\_FILES['myfile']['error']-ს აქვს 5 მნიშვნელობა:

0 - უშეცდომოდ, ფაილი წარმატებით ჩაიტვირთა

1 - ჩასატვირთი ფაილი აღემატება იმ ზომას, რომელიც განსაზღვრულია *php.ini* ფაილის მოწყობის დირექტივით upload\_max\_filesize

2 - ჩასატვირთი ფაილი აღემატება იმ ზომას, რომელიც განსაზღვრულია html ფორმის MAX\_FILE\_SIZE ელემენტით

3 - ფაილი ნაწილობრივ ჩაიტვირთა

4 - ფაილი არ ჩაიტვირთა

ჩუმათობის პრინციპით, ჩასატვირთი ფაილი მოთავსდება სერვერის დროებით დირექტორიაში, თუ კი არაა მითითებული სხვა დირექტორია ფაილის მოწყობის *php.ini* ოპციაში - upload\_tmp\_dir. ჩატვირთული ფაილის საჭირო დირექტორიაში მოთავსება შესაძლებელია move\_uploaded\_file() ფაილის საშუალებით.

*ფუნქცია*  move\_uploaded\_file()-ს სინტაქსისია:

bool move\_uploaded\_file (ფაილის დროებითი სახელი, დანიშნულების ადგილი)

ეს ფუნქცია ამოწმენს, მართლაც ჩაიტვირთა თუ არა მითითებული ფაილი, *POST* მეთოდით, *HTTP* ჩატვირთვის მექანიზმის გამოყენებით, თუ ეს ასეა, მაშინ ცატვირთული ფაილი გადაიტანება მითითებულ ადგილას (ეს პარამეტრი შეიცავს ადგილის გზასთან ერთად, ფაილის ახალ სახელს).

თუ კი მითითებული იქნება არ ჩატვირთული ფაილის დროებითი სახელი, მაშინ move\_uploaded\_file() ფუნქცია დააბრუნებს  FALSE-ს. იგივე შედეგი იქნება, თუ კი რაიმე მიზეზის გამო ფაილის გადატანა არ მოხდა. ამ შემთხვევაში ინტერპრეტატორი შესაბამის შეტყობინებას გამოიტანს. თუ კი მითითებულ ადგილას არის იგივე სახელის ფაილი მაშინ ფუნქცია move\_uploaded\_file() მას გადააწერს ახალ ფაილს.

ლისტინგი 9. ჩატვირთული ფაილის გადატანა

<?

/\* PHP -ს 4.1.0 ვერსიებზე უფრო ადრინდელებში

$\_FILES მასივის ნაცვლად გამოიყენება

მასივი $HTTP\_POST\_FILES \*/

$uploaddir = 'c:/uploads/';

// ჩასატვირთ ფაილს შევინახავთ

// ამ დირექტორიაში

$destination = $uploaddir .

$\_FILES['myfile']['name'];

// ფაილის სახელი არ შევცვალოთ

print "<pre>";

if (move\_uploaded\_file(

$\_FILES['myfile']['tmp\_name'],

$destination)) {

/\* ფაილს გადავიტანთ დროებითი ფოლდერიდან

ამორჩეულ დირექტორიაში \*/

print "ფაილი წარმატებით ჩაიტვირთა <br>";

} else {

echo "მოხდა შეცდომა ჩატვირთვისას.

ზოგიერთი ინფორმაცია გამართვისთვის:<br>";

print\_r($\_FILES);

}

print "</pre>";

?>