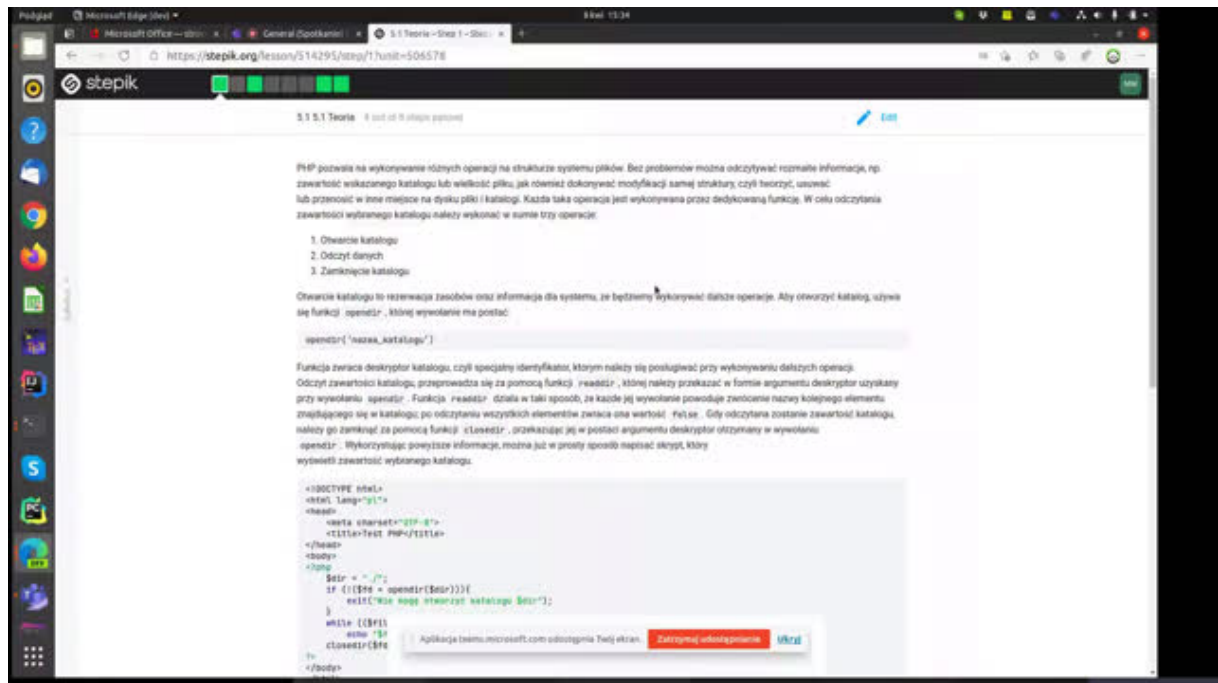


10.2 Teoria

Step 1



To watch this video please visit <https://stepik.org/lesson/514295/step/1>

Step 2

PHP pozwala na wykonywanie różnych operacji na strukturze systemu plików. Bez problemów można odczytywać rozmaite informacje, np. zawartość wskazanego katalogu lub wielkość pliku, jak również dokonywać modyfikacji samej struktury, czyli tworzyć, usuwać lub przenosić w inne miejsce na dysku pliki i katalogi. Każda taka operacja jest wykonywana przez dedykowaną funkcję. W celu odczytania zawartości wybranego katalogu należy wykonać w sumie trzy operacje:

1. Otwarcie katalogu
2. Odczyt danych
3. Zamknięcie katalogu

Otwarcie katalogu to rezerwacja zasobów oraz informacja dla systemu, że będziemy wykonywać dalsze operacje. Aby otworzyć katalog, używa się funkcji `opendir`, której wywołanie ma postać:

```
opendir('nazwa_katalogu')
```

Funkcja zwraca deskryptor katalogu, czyli specjalny identyfikator, którym należy się posługiwać przy wykonywaniu dalszych operacji. Odczyt zawartości katalogu, przeprowadza się za pomocą funkcji `readdir`, której należy przekazać w formie argumentu deskryptor uzyskany przy wywołaniu `opendir`. Funkcja `readdir` działa w taki sposób, że każde jej wywołanie powoduje zwrócenie nazwy kolejnego elementu znajdującego się w katalogu; po odczytaniu wszystkich elementów zwraca ona wartość `false`. Gdy odczytana zostanie zawartość katalogu, należy go zamknąć za pomocą funkcji `closedir`, przekazując jej w postaci argumentu deskryptor otrzymany w wywołaniu `opendir`. Wykorzystując powyższe informacje, można już w prosty sposób napisać skrypt, który wyświetli zawartość wybranego katalogu.

```
<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="UTF-8">
  <title>Test PHP</title>
</head>
<body>
<?php
  $dir = "./";
  if (!($fd = opendir($dir))){
    exit("Nie mogę otworzyć katalogu $dir");
  }
  while (($file = readdir($fd)) !== false)
    echo "$file<br/>";
  closedir($fd);
?>
</body>
</html>
```

W PHP5 istnieje również funkcja o nazwie `scandir`, która za jednym wywołaniem pobiera zawartość całego katalogu i zwraca ją w postaci tablicy. Wywołanie tej funkcji ma postać:

```
scandir('nazwa katalogu'[, sortowanie])
```

Parametr `nazwa_katalogu` określa katalog, z którego będą pobierane dane, natomiast `sortowanie` – sposób sortowania.

Ustawienie drugiego parametru na `0` powoduje, że nazwy plików i katalogów będą sortowane alfabetycznie w porządku rosnącym, a ustawienie na `1` lub inną wartość niezerową – że nazwy będą sortowane w porządku malejącym.

```
<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="UTF-8">
  <title>Test PHP</title>
</head>
<body>
<?php
  $dir = "./";
  $arr = scandir("$dir");
  foreach ($arr as $file){
    if($file != '.' && $file != '..')
      echo "$file<br/>";
  }
?>
</body>
</html>
```

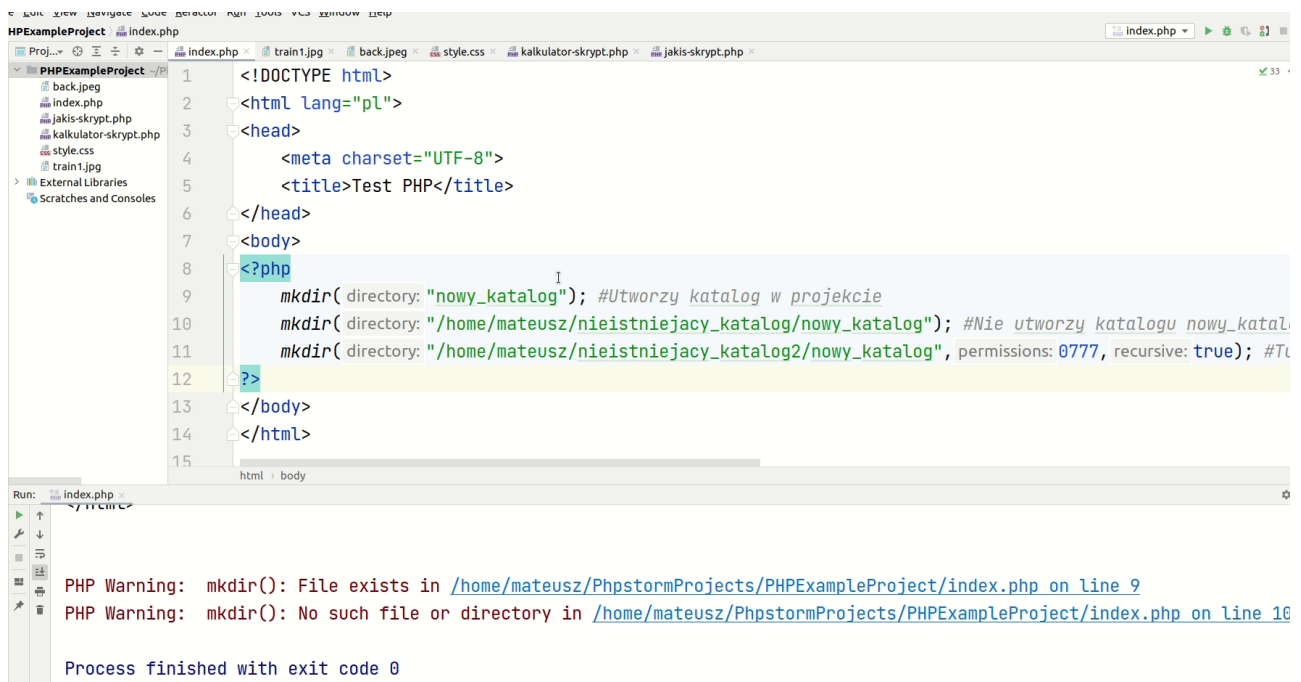
Step 3

Do tworzenia katalogów służy funkcja `mkdir`. Jej wywołanie wygląda następująco:

```
mkdir('nazwa'[,tryb[, zagnieżdżone]])
```

Parametr `nazwa` określa tu nazwę katalogu, natomiast `tryb` – prawa dostępu. Argument `nazwa` może określać zarówno względną, jak i bezwzględną ścieżkę dostępu. Jeśli nie ma określonej ścieżki, nowy katalog zostanie utworzony w katalogu bieżącym. Trzeci parametr ustawiony na `true` umożliwia utworzenie zagnieżdżonej struktury katalogów.

```
<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="UTF-8">
  <title>Test PHP</title>
</head>
<body>
<?php
  mkdir("nowy_katalog"); #Utworzy katalog w projekcie
  mkdir("/home/mateusz/nieistniejący_katalog/nowy_katalog"); #Nie utworzy katalogu nowy_katalog, ponieważ
  nie ma katalogu o nazwie nieistniejący katalog
  mkdir("/home/mateusz/nieistniejący_katalog2/nowy_katalog",0777,true); #Tutaj zostanie utworzona ścieżka
  katalogu oraz zostanie jemu przydzielone pełne prawa dostępu w systemie Linux
?>
</body>
</html>
```



```
<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="UTF-8">
  <title>Test PHP</title>
</head>
<body>
<?php
    mkdir( directory: "nowy_katalog"); #Utworzy katalog w projekcie
    mkdir( directory: "/home/mateusz/nieistniejacy_katalog/nowy_katalog"); #Nie utworzy katalogu nowy_katal
    mkdir( directory: "/home/mateusz/nieistniejacy_katalog2/nowy_katalog", permissions: 0777, recursive: true); #T
  ?>
</body>
</html>
```

Run: index.php

PHP Warning: mkdir(): File exists in /home/mateusz/PhpstormProjects/PHPExampleProject/index.php on line 9

PHP Warning: mkdir(): No such file or directory in /home/mateusz/PhpstormProjects/PHPExampleProject/index.php on line 10

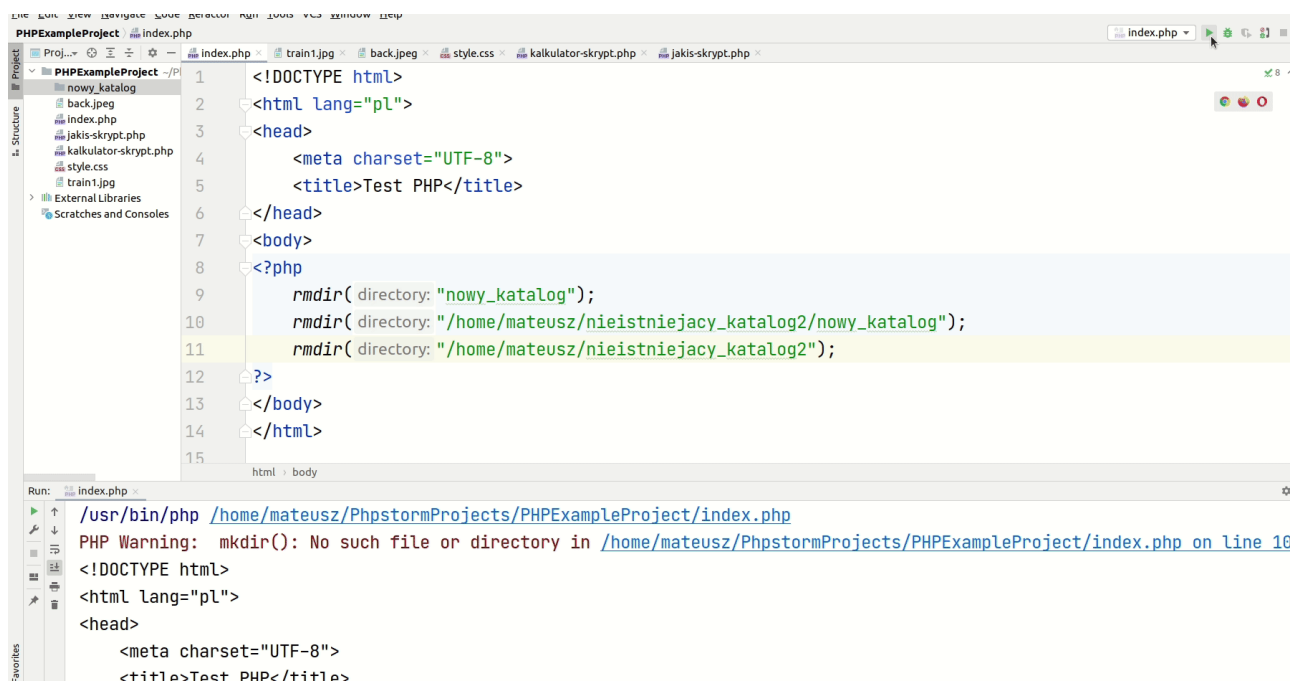
Process finished with exit code 0

Do usuwania katalogów służy funkcja `rmdir`. Jego wywołanie wygląda następująco:

```
rmdir('nazwa')
```

Ważne, że usuwany katalog **musi być pusty**.

```
<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="UTF-8">
  <title>Test PHP</title>
</head>
<body>
<?php
    rmdir("nowy_katalog");
    rmdir("/home/mateusz/nieistniejacy_katalog2/nowy_katalog");
    rmdir("/home/mateusz/nieistniejacy_katalog2");
?>
</body>
</html>
```



```
<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="UTF-8">
  <title>Test PHP</title>
</head>
<body>
<?php
    rmdir( directory: "nowy_katalog");
    rmdir( directory: "/home/mateusz/nieistniejacy_katalog2/nowy_katalog");
    rmdir( directory: "/home/mateusz/nieistniejacy_katalog2");
  ?>
</body>
</html>
```

Run: index.php

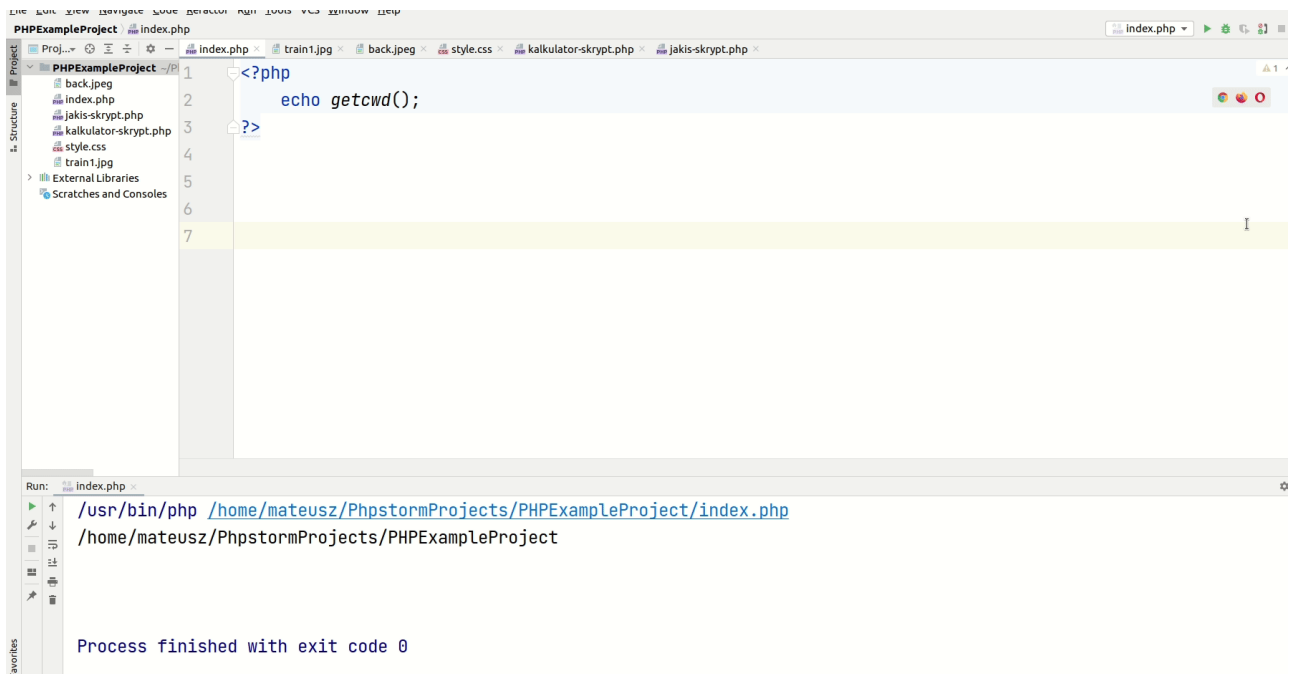
/usr/bin/php /home/mateusz/PhpstormProjects/PHPExampleProject/index.php

PHP Warning: mkdir(): No such file or directory in /home/mateusz/PhpstormProjects/PHPExampleProject/index.php on line 10

```
<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="UTF-8">
  <title>Test PHP</title>
```

Jeśli chcemy się dowiedzieć, jaki jest aktualny katalog bieżący, powinniśmy użyć funkcji `getcwd`, która zwraca jego nazwę w postaci ciągu znaków.

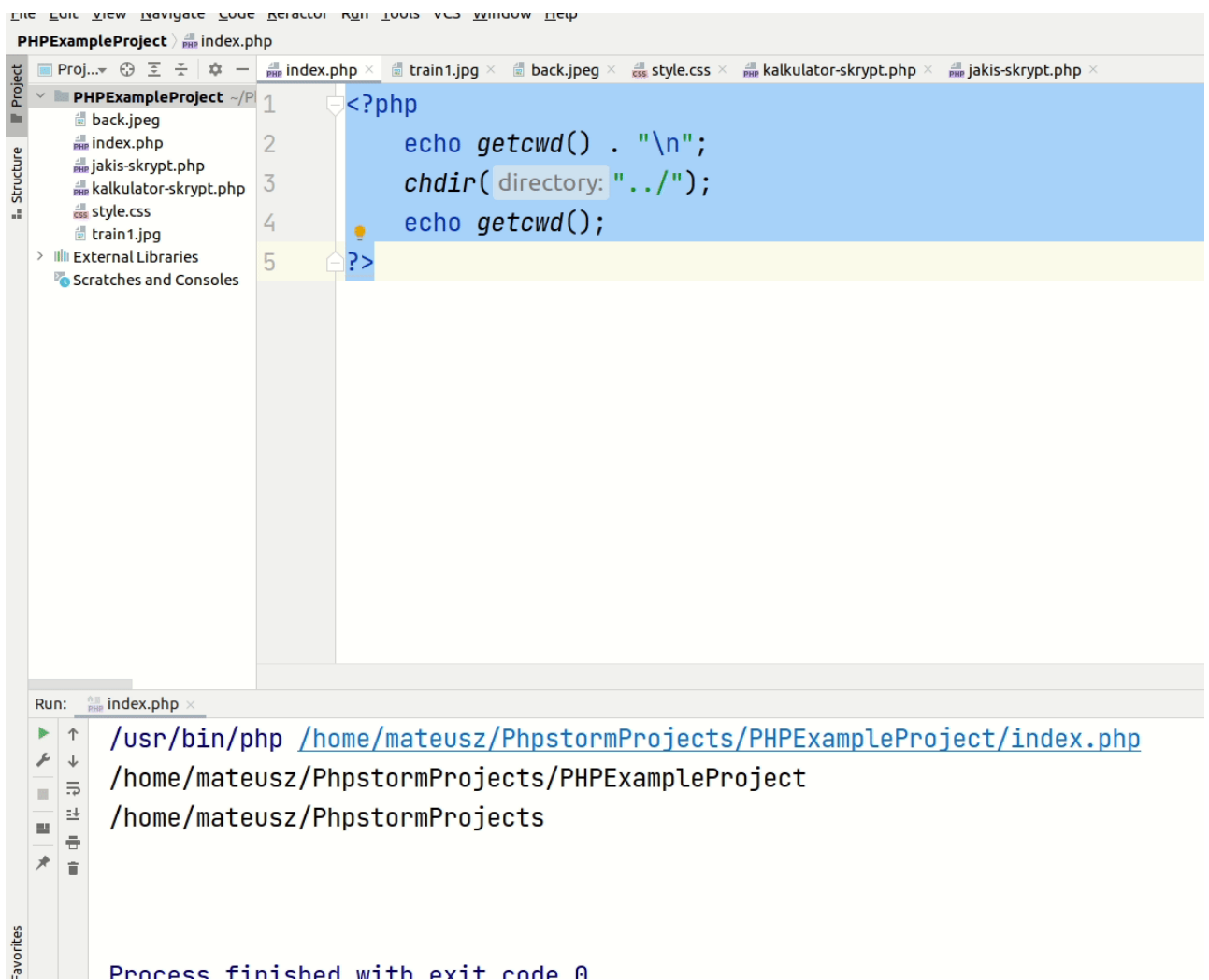
```
<?php
    echo getcwd();
?>
```



Do zmiany katalogu bieżącego wykorzystywana jest funkcja `chdir`, której w postaci argumentu należy przekazać nazwę nowego katalogu bieżącego. Wywołanie ma postać:

```
chdir('nazwa')
```

```
<?php
    echo getcwd() . "\n";
    chdir("../");
    echo getcwd();
?>
```

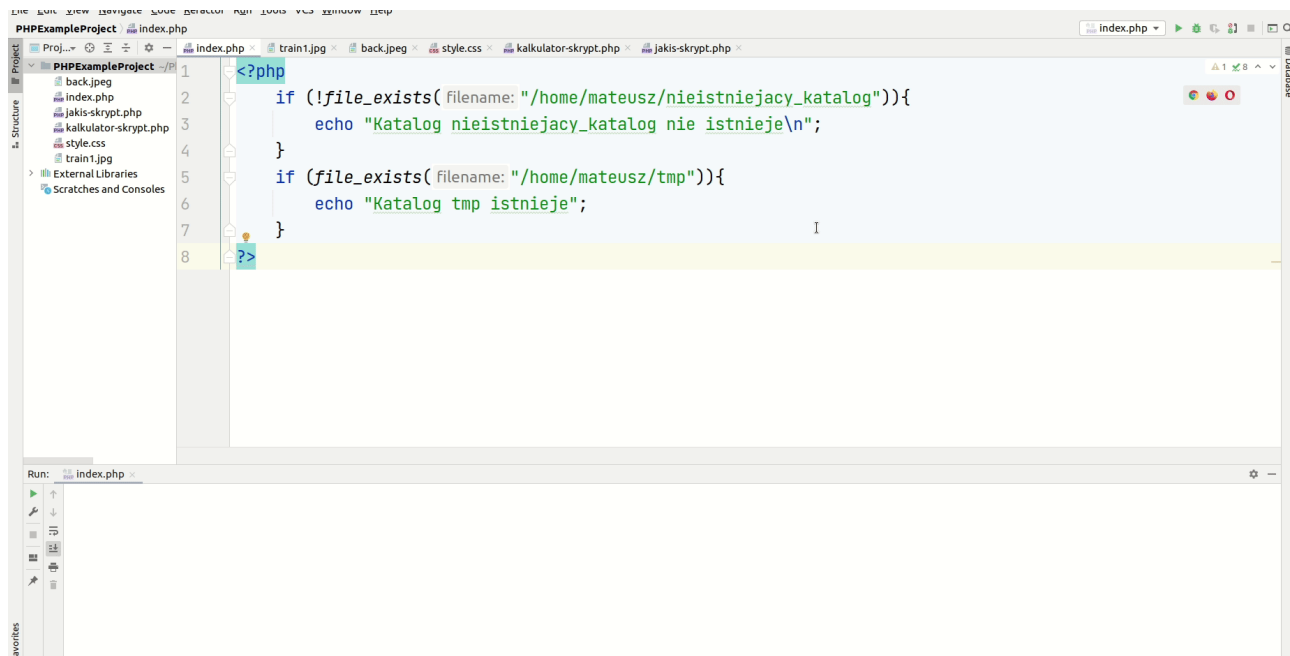


Aby ustalić czy dany katalog istnieje należy użyć funkcji `file_exists`, której wywołanie ma postać:

```
file_exists('nazwakatalogu')
```

Zwracaną wartością jest `true`, jeśli katalog istnieje, lub `false` w przeciwnym razie. Ta sama funkcja jest wykorzystywana do sprawdzania plików.

```
<?php
if (!file_exists("/home/mateusz/nieistniejacy_katalog")){
    echo "Katalog nieistniejacy_katalog nie istnieje\n";
}
if (file_exists("/home/mateusz/tmp")){
    echo "Katalog tmp istnieje";
}
?>
```



Step 4

Czynności tworzenia plików na dysku oraz ich otwierania są ze sobą związane i wykonuje je jedna funkcja – `fopen`. Wywołanie funkcji ma postać:

```
fopen ( string$nazwa_pliku , string$tryb [, bool$include_path ] )
```

Argument `nazwa_pliku` to ciąg znaków wskazujący nazwę pliku, który należy otworzyć. Parametr `tryb_otwarcia` określa tryb otwarcia pliku i może przyjmować wartości:

Tryb	Opis
'r'	Otwiera plik tylko do odczytu; umieszcza wskaźnik pliku na jego początku.
'r+'	Otwiera plik do odczytu i zapisu; umieszcza wskaźnik pliku na jego początku.
'w'	Otwiera plik tylko do zapisu; umieszcza wskaźnik pliku na jego początku i obcina plik do zerowej długości. Jeśli plik nie istnieje, to próbuje go utworzyć.
'w+'	Otwiera plik do odczytu i zapisu; umieszcza wskaźnik pliku na jego początku i obcina plik do zerowej długości. Jeśli plik nie istnieje, to próbuje go utworzyć.
'a'	Otwiera plik tylko do zapisu; umieszcza wskaźnik pliku na jego końcu. Jeśli plik nie istnieje, to próbuje go utworzyć.
'a+'	Otwiera plik do odczytu i zapisu; umieszcza wskaźnik pliku na jego końcu. Jeśli plik nie istnieje, to próbuje go utworzyć.
'x'	Tworzy i otwiera plik tylko do zapisu; umieszcza wskaźnik pliku na jego początku. Jeśli plik już istnieje, wywołanie funkcji fopen() nie powiedzie się.
'x+'	Tworzy i otwiera plik do odczytu i zapisu; umieszcza wskaźnik pliku na jego początku. Jeśli plik już istnieje, wywołanie fopen() nie powiedzie się.

Parametr `tryb_otwarcia` może również zawierać określenie typu pliku: `b` – dla plików binarnych bądź `t` – dla plików tekstowych.

Parametr `include_path` jest to opcjonalny parametr, który może być ustawiony na `'1'` lub `true` jeśli chcesz szukać pliku także w `include_path`.

Gdy zakończą się wykonywane na pliku operacje, powinien on zostać zamknięty za pomocą funkcji `fclose`, której typowe wywołanie ma postać

```
fclose(deskryptor);
```

`deskryptor` to oczywiście wartość uprzednio zwrócona przez `fopen`.

Pojedyncze wiersze tekstu można odczytać z pliku za pomocą funkcji o nazwie `fgets`, której schematyczne wywołanie ma postać:

```
fgets(deskryptor[, ile])
```

`deskryptor` to oczywiście wartość uprzednio zwrócona przez `fopen`, a argument `ile` jest opcjonalny i określa maksymalną liczbę znaków do odczytu. Funkcję `fgets` najczęściej stosuje się w pętli `While` testującej osiągnięcie końca pliku (`feof`). Pokazuje to poniższy przykład:

```
<?php
    if (!$fd = fopen('./test.txt', 'r')){
        echo "Nie można otworzyć pliku test.txt";
    }
    else{
        while(!feof($fd)){
            $str = fgets($fd);
            $str = nl2br($str); #Funkcja ta dodaje automatycznie znacznik <br/>
            echo $str;
        }
        fclose($fd);
    }
?>
```

Step 5

Jeżeli spodziewamy się, że w pliku, z którego będziemy odczytywać dane, znajdują się niepotrzebne nam znaczniki HTML, możemy do odczytu danych wykorzystać funkcję `fgetss`. Jej wywołanie ma postać:

```
fgetss(deskryptor[, ile[, tagi]])
```

`deskryptor` to oczywiście wartość uprzednio zwrócona przez `fopen`, argument `ile` jest opcjonalny i określa maksymalną liczbę znaków do odczytu, a parametr `tagi` określa znaczniki, które mają nie być usuwane.

```
<?php
    if (!$fd = fopen('plik.html', 'r')){
        echo "Nie można otworzyć pliku plik.html";
    }
    else{
        $lineno = 0;
        while(!feof($fd)){
            $str = fgetss($fd);
            $str = nl2br($str); #Funkcja ta dodaje automatycznie znacznik <br/>
            echo $lineno . " $str";
            $lineno++;
        }
        fclose($fd);
    }
?>
```

Uwaga. Funkcja `fgetss` nie występuje już w PHP8 i jej używanie jest wysoko odradzane.

Odczyt pojedynczych znaków z pliku umożliwia funkcja `fgetc`. Przyjmuje ona jeden argument, jakim jest deskryptor pliku, który będzie odczytywany, oraz zwraca ciąg zawierający pojedynczy odczytany znak. Wywołanie ma więc schematyczną postać:

```
fgetc(deskryptor);
```

Po jej wykonaniu wskaźnik pliku przesuwany jest o jeden bajt do przodu. W przypadku gdy zostanie osiągnięty koniec pliku (`feof`), `fgetc` zwraca wartość `false`.

```
<?php
    if (!$fd = fopen('test.txt', 'r')){
        echo "Nie można otworzyć pliku test.txt";
    }
    else{
        while (($str = fgetc($fd)) !== false){
            if ($str == "\n")
                $str = "<br/>";
            echo $str;
        }
        fclose($fd);
    }
?>
```

Kiedy chce się odczytać określoną liczbę bajtów lub też dane z pliku binarnego, należy użyć funkcji `fread`. Jej wywołanie ma postać:

```
fread(deskryptor, ile)
```

`deskryptor` to wartość zwrócona przez `fopen`, argument `ile` określa liczbę bajtów do odczytania. Funkcja zwraca odczytany fragment pliku w postaci ciągu typu string.

```
<?php
    if (!$fd = fopen('test.txt', 'rb')){
        echo "Nie można otworzyć pliku test.txt";
    }
    else{
        while(!feof($fd)){
            echo fread($fd, 4096);
        }
        fclose($fd);
    }
?>
```

Powyższy kod można przerobić tak, aby nie korzystać z funkcji `feof` następująco:

```
<?php
    if (!$fd = fopen('test.txt', 'rb')){
        echo "Nie można otworzyć pliku test.txt";
    }
    else{
        while (($str = fread($fd, 4096)) !== ""){
            echo $str;
        }
        fclose($fd);
    }
?>
```

Odczyt pełnej zawartości pliku można wykonać na kilka sposobów. Sposób pierwszy to wykorzystanie przedstawionej wyżej funkcji `fread`, której jako drugi argument zostanie przekazana wielkość pliku (funkcja `filesize`).

```
<?php
    if (!$fd = fopen('test.txt', 'rb')){
        echo "Nie można otworzyć pliku test.txt";
    }
    else{
        $r = filesize("test.txt");
        $str = fread($fd,$r);
        echo $str;
        fclose($fd);
    }
?>
```

Czynność taka może również zostać przeprowadzona dużo prościej, jeśli skorzysta się z funkcji dedykowanych `readfile` lub `file_get_contents`. Aby wysłać zawartość pliku `test.txt` do przeglądarki, wystarczy wykonać tylko jedną instrukcję:

```
readfile('test.txt');
```

W podobny sposób działa też `file_get_contents`. Jako argument przyjmuje ona nazwę pliku, zwraca natomiast jego odczytaną zawartość w postaci wartości typu string lub wartości `false`, jeśli odczyt się nie udał.

```
<?php
if (!$fd = fopen('test.txt', 'rb')){
    echo "Nie można otworzyć pliku test.txt";
}
else{
    echo readfile("test.txt") . "<br/>";
    echo file_get_contents("test.txt");
}
?>
```

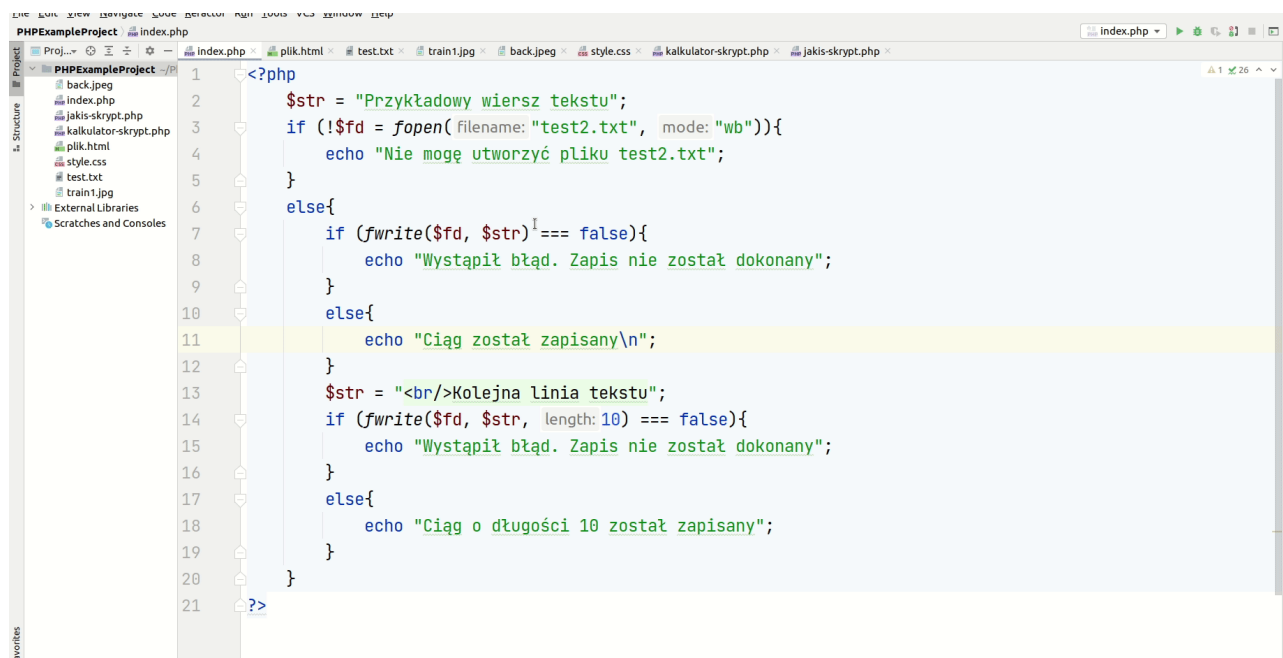
Step 6

Zapis danych do pliku może być przeprowadzony za pomocą funkcji `fwrite` (można też korzystać z jej aliasu `fputs`). Przyjmuje ona trzy argumenty, a jej schematyczne wywołanie ma postać:

```
fwrite(deskryptor, str[, ile])
```

`deskryptor` to deskryptor pliku zwrócony przez wywołanie funkcji `fopen`. `str` to ciąg typu string zawierający dane, które mają zostać zapisane, natomiast `ile` to opcjonalny parametr wskazujący, ile bajtów z ciągu `str` ma zostać zapisanych.

```
<?php
$str = "Przykładowy wiersz tekstu";
if (!$fd = fopen("test2.txt", "wb")){
    echo "Nie mogę utworzyć pliku test2.txt";
}
else{
    if (fwrite($fd, $str) === false){
        echo "Wystąpił błąd. Zapis nie został dokonany";
    }
    else{
        echo "Ciąg został zapisany\n";
    }
    $str = "<br/>Kolejna linia tekstu";
    if (fwrite($fd, $str, 10) === false){
        echo "Wystąpił błąd. Zapis nie został dokonany";
    }
    else{
        echo "Ciąg o długości 10 został zapisany";
    }
}
?>
```



Drugą funkcją, która pozwala na zapis danych do pliku, jest `file_put_contents`. Ta funkcja może przyjmować do czterech argumentów. Schemat jej wywołania jest następujący:

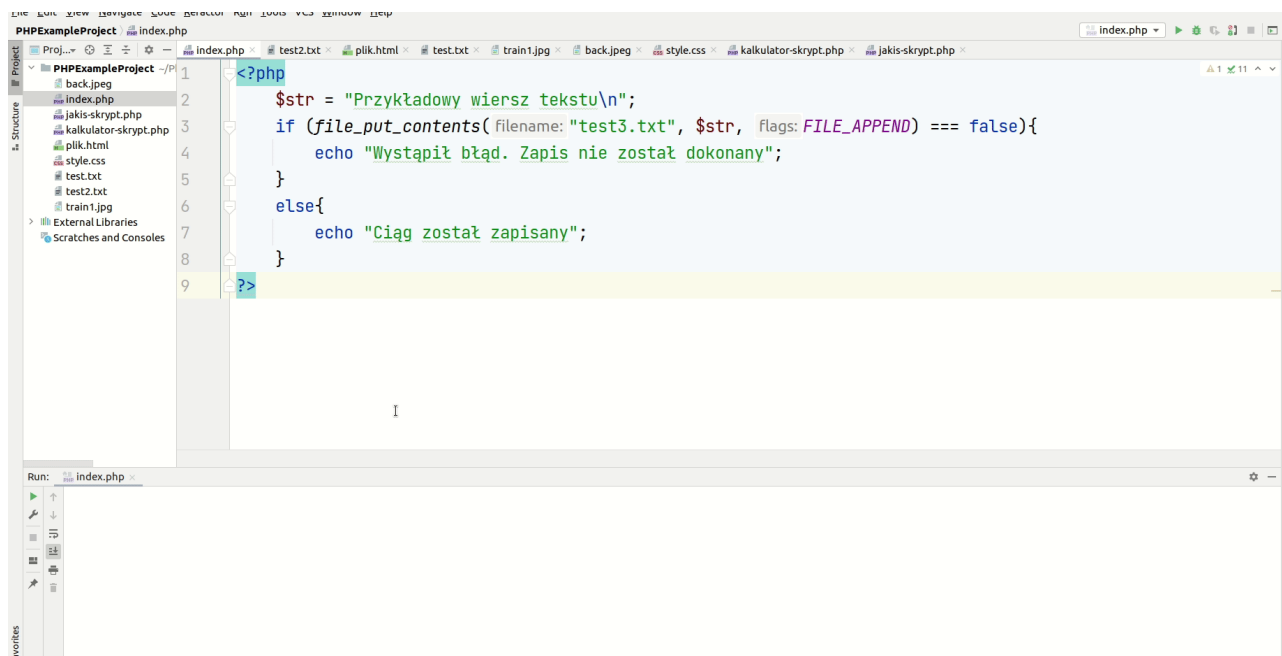
```
file_put_contents(„nazw_pliku”, dane[, flagi]);
```

Oznacza ono zapisanie danych wskazywanych przez argument `dane` do pliku o nazwie wskazywanej przez argument `nazwa_pliku`. Argument `dane` może być ciągiem znaków bądź tablicą. Parametr `flagi` może przyjmować następujące wartości;

- `FILE_USE_INCLUDE_PATH` – obecność pliku będzie sprawdzana w lokalizacjach wskazanych przez zmienną konfiguracyjną `include_path`.

- `FILE_APPEND` – dane będą dopisywane na końcu pliku
- `LOCK_EX` – plik zostanie zablokowany na czas zapisu.

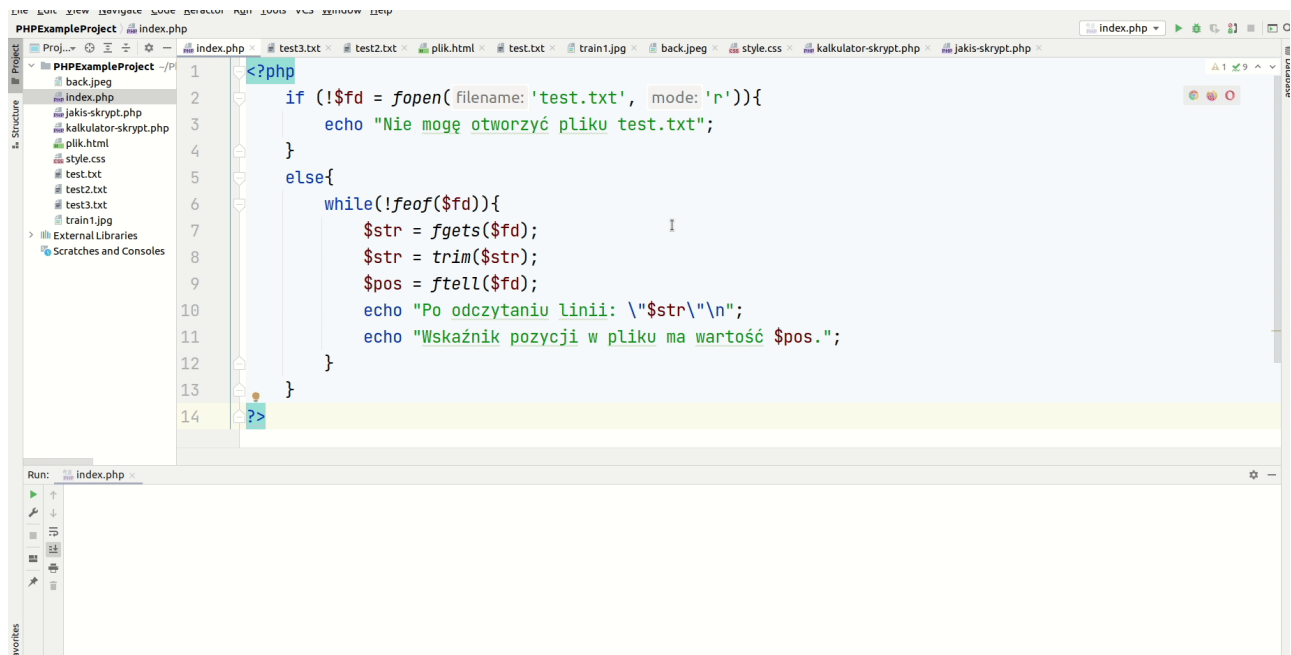
```
<?php
$str = "Przykładowy wiersz tekstu\n";
if (file_put_contents("test3.txt", $str, FILE_APPEND) === false){
    echo "Wystąpił błąd. Zapis nie został dokonany";
}
else{
    echo "Ciąg został zapisany";
}
?>
```



Step 7

Odczyt danych powoduje przesuwanie wskaźnika pozycji w pliku, dzieje się to jednak automatycznie, bez wiedzy programisty. Istnieje jednak możliwość odczytania aktualnej pozycji tego wskaźnika, a także zmiany jego wartości. Służą do tego trzy funkcje: `ftell`, `fseek` i `rewind`. Funkcja `ftell` zwraca aktualną pozycję w pliku (typ `int`). Przyjmuje ona jako argument deskryptor pliku otwartego za pomocą `fopen`.

```
<?php
if (!$fd = fopen('test.txt', 'r')){
    echo "Nie mogę otworzyć pliku test.txt";
}
else{
    while(!feof($fd)){
        $str = fgets($fd);
        $str = trim($str);
        $pos = ftell($fd);
        echo "Po odczytaniu linii: \"$str\"\n";
        echo "Wskaźnik pozycji w pliku ma wartość $pos.";
    }
}
?>
```



Aktualna pozycja w pliku może zostać zmieniona za pomocą funkcji `fseek`. Jej schematyczne wywołanie ma postać:

```
fseek(deskryptor, ile[, skąd])
```

`deskryptor` to parametr zwrócony przez funkcję `fopen`, `ile` określa liczbę bajtów przesunięcia, natomiast `skąd` określa pozycję, od której nastąpi przesunięcie.

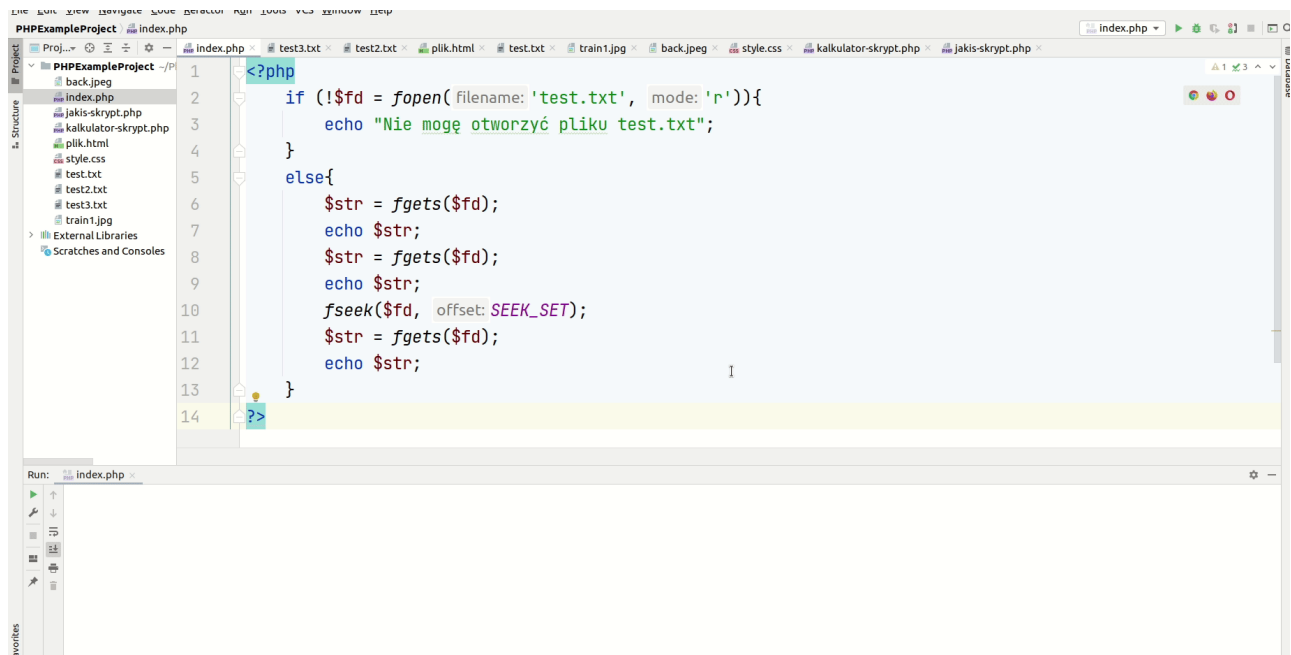
Parametr `skąd` jest opcjonalny i może przyjmować następujące wartości:

- `SEEK_SET` – oznacza przesunięcie względem początku pliku
- `SEEK_CUR` – oznacza przesunięcie względem pozycji bieżącej
- `SEEK_END` – oznacza przesunięcie względem końca pliku

Domyślną wartością jest `SEEK_SET`. Jeśli na przykład zmienna `$fd` zawiera deskryptor pliku zwrócony przez `fopen` to wywołanie:

- `fseek($fd, 0)` – ustawi wskaźnik na początku pliku
- `fseek($fd, 0, SEEK_SET)` – ustawi wskaźnik na początku pliku
- `fseek($fd, 0, SEEK_END)` – ustawi wskaźnik na końcu pliku
- `fseek($fd, 10, SEEK_SET)` – ustawi wskaźnik na 20 bajcie pliku
- `fseek($fd, -10, SEEK_CUR)` – ustawi wskaźnik na 10 bajcie przed aktualną pozycją pliku
- `fseek($fd, 10, SEEK_CUR)` – ustawi wskaźnik na 10 bajcie za aktualną pozycją pliku
- `fseek($fd, -10, SEEK_END)` – ustawi wskaźnik na 10 bajcie przed końcem pliku

```
<?php
if (!$fd = fopen('test.txt', 'r')){
    echo "Nie mogę otworzyć pliku test.txt";
}
else{
    $str = fgets($fd);
    echo $str;
    $str = fgets($fd);
    echo $str;
    fseek($fd, SEEK_SET);
    $str = fgets($fd);
    echo $str;
}
?>
```

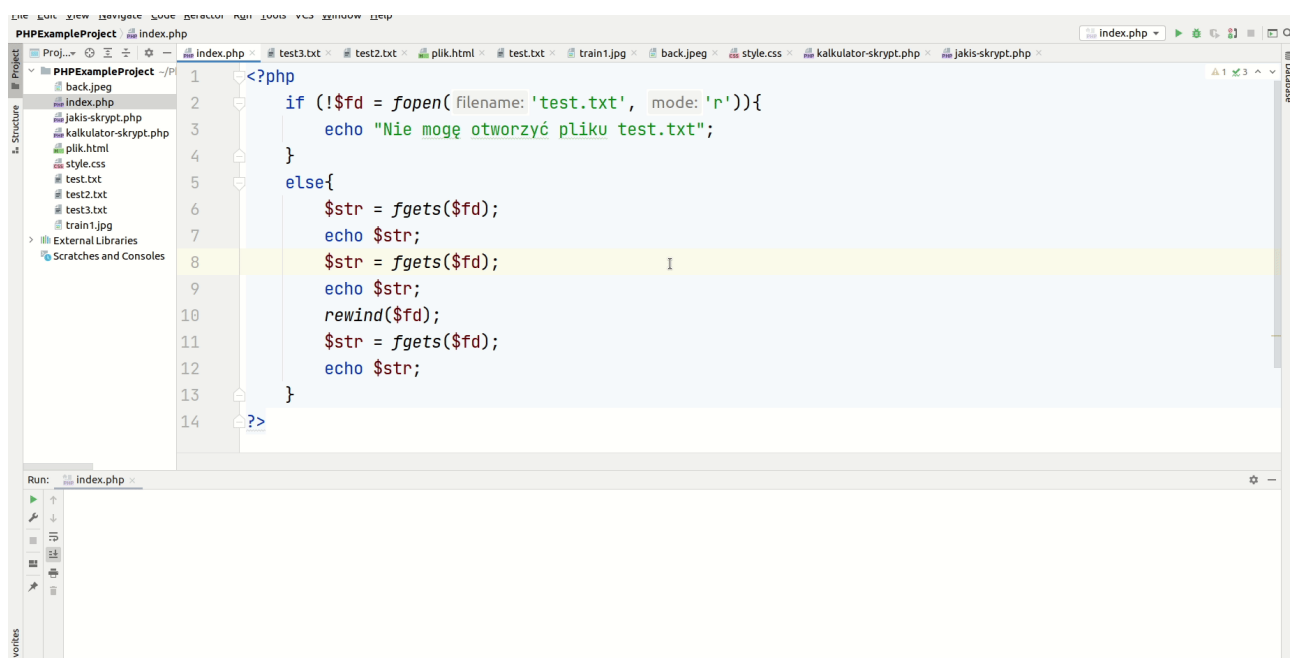


Ostatnia z omawianych funkcji – `rewind` – przesuwa wskaźnik pozycji w pliku na pozycję 0, czyli na jego początek. Funkcja `rewind` przyjmuje tylko jeden argument, którym jest deskryptor pliku.

```

<?php
if (!$fd = fopen('test.txt', 'r')){
    echo "Nie mogę otworzyć pliku test.txt";
}
else{
    $str = fgetc($fd);
    echo $str;
    $str = fgetc($fd);
    echo $str;
    rewind($fd);
    $str = fgetc($fd);
    echo $str;
}
?>

```



Step 8

Jeżeli nie zadamy o prawidłową synchronizację dostępu przechowywanie danych w plikach może przysporzyć nam wielu problemów. Co się bowiem stanie, jeśli naszą witrynę odwiedzi naraz kilka osób i wszystkie wywołują skrypt operujący na danych zapisanych w jednym z plików? Do synchronizacji dostępu do pliku można użyć funkcji `flock`. Wywołanie funkcji `flock` powoduje założenie takiej blokady na dany plik, że dostęp do niego będzie miał tylko skrypt, który tę funkcję wywołał. Wywołanie ma postać:

```
flock(deskryptor, operacja)
```

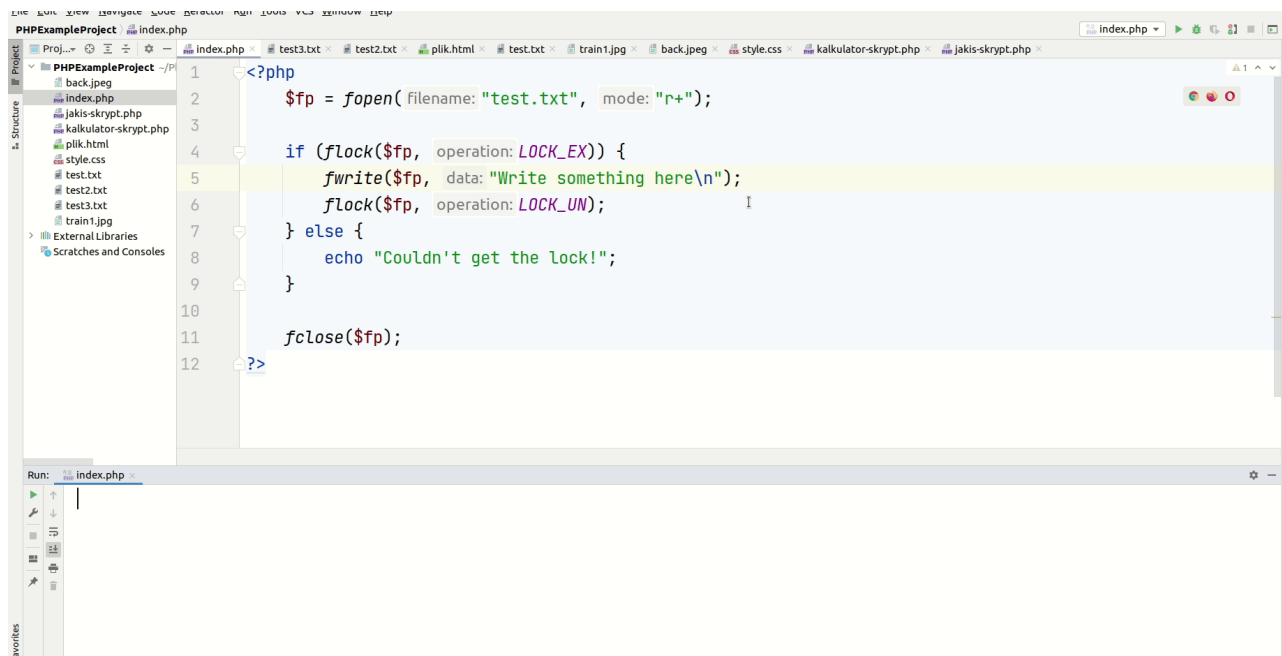
deskryptor to parametr zwrócony przez funkcję `fopen`, natomiast `operacja` określa rodzaj blokowania. Parametr ten może przyjmować następujące wartości:

- `LOCK_SH` – blokada zapisu, możliwy odczyt.
- `LOCK_EX` – pełna blokada.
- `LOCK_UN` – zwolnienie blokady.

```
<?php
$fp = fopen("test.txt", "r+");

if (flock($fp, LOCK_EX)) {
    fwrite($fp, "Write something here\n");
    flock($fp, LOCK_UN);
} else {
    echo "Couldn't get the lock!";
}

fclose($fp);
?>
```

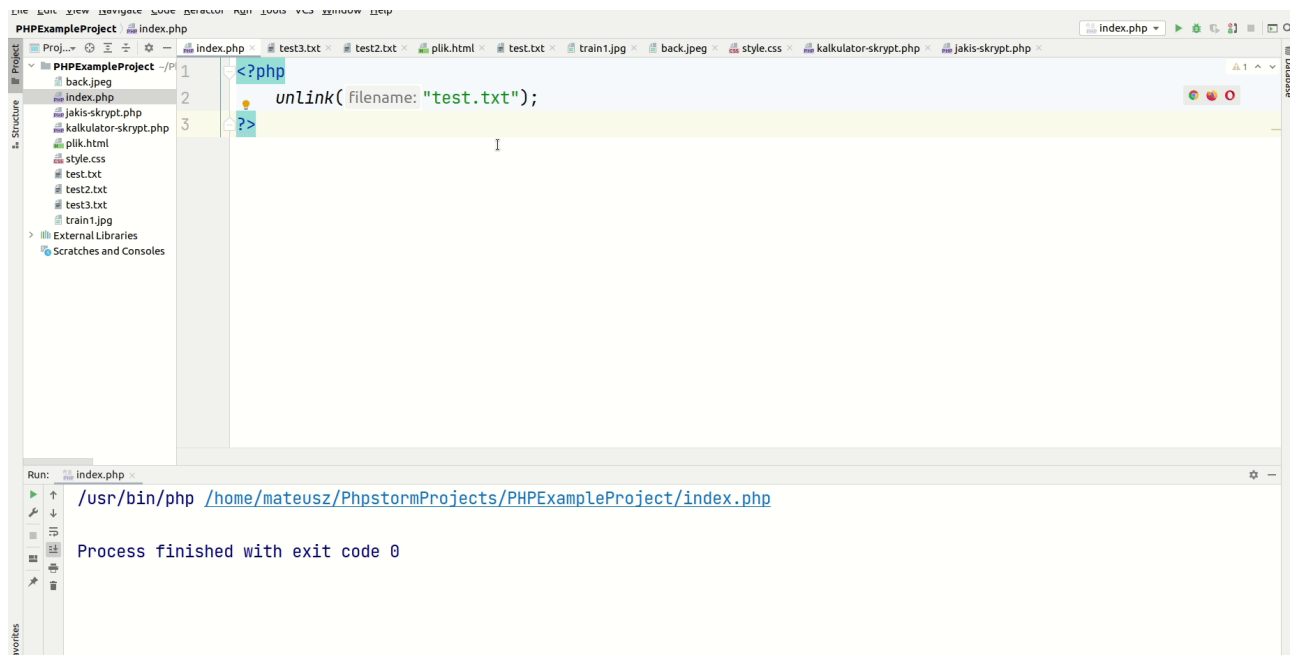


Step 9

Do usunięcia plików służy funkcja `unlink`, która w postaci argumentu przyjmuje nazwę usuwanego pliku, a zatem jej wywołanie ma schematyczną postać:

```
unlink('nazwa_pliku');
```

```
<?php
    unlink("test.txt");
?>
```



Rozmiar pliku można sprawdzić, wywołując funkcję o nazwie `filesize`, schematycznie:

```
filesize('nazwapliku')
```

Funkcja zwraca wartość typu `integer` określającą wielkość pliku w bajtach.

Oprócz informacji o wielkości pliku przydatnych może być również kilka innych danych pobieranych za pomocą odpowiednich funkcji. Oto niektóre z nich:

- `fileatime` – zwraca znacznik czasu Uniksa określający czas ostatniego dostępu do pliku
- `filectime` – zwraca znacznik czasu Uniksa określający czas ostatniej zmiany metadanych pliku.
- `filetype` – zwraca ciąg znaków określający typ pliku. Zwracane wartości to: `fifo`, `char`, `dir`, `block`, `link`, `file` i `unknown`.
- `fileperms` – zwraca wartość określającą prawa dostępu do pliku
- `fileowner` – zwraca identyfikator właściciela pliku.

```
<?php
echo filesize('test2.txt') . "\n";
echo filetype('test2.txt') . "\n";
echo filetype('../') . "\n";
echo fileperms('test2.txt') . "\n";
echo fileowner('test2.txt') . "\n";
?>
```

Jeśli chcemy uzyskać informacje o ilości wolnego miejsca na dysku, możemy skorzystać z funkcji `disk_free_space`. Przyjmuje ona w postaci argumentu `nazwę katalogu` i zwraca ilość wolnego miejsca (w bajtach) na dysku logicznym, na którym znajduje się ten katalog. Jeśli interesuje nas nie wolna, ale całkowita ilość miejsca, zamiast `disk_free_space` należy zastosować funkcję `disk_total_space`.

```
<?php
$wolne_miejsce = disk_free_space("/");
$calkowite_miejsce = disk_total_space("/");
$zajete_miejsce = $calkowite_miejsce - $wolne_miejsce;

echo "Całkowite miejsce na dysku: $calkowite_miejsce\n";
echo "Dostępne miejsce na dysku: $wolne_miejsce\n";
echo "Zajęte miejsce na dysku: $zajete_miejsce\n";
?>
```