

# Operacje na plikach

## 1. Tworzenie i otwieranie plików

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\$użyj\_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:

### Lista możliwych trybów dla fopen()

tryb	Opis
'r'	Otwiera tylko do odczytu; umieszcza wskaźnik pliku na jego początku.
'r+'	Otwiera do odczytu i zapisu; umieszcza wskaźnik pliku na jego początku.
'w'	Otwiera 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 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 tylko do zapisu; umieszcza wskaźnik pliku na jego końcu. Jeśli plik nie istnieje to próbuje go utworzyć.
'a+'	Otwiera 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 <b>fopen()</b> nie powiedzie się, zwróci <b>FALSE</b> i wygeneruje błąd.
'x+'	Tworzy i otwiera plik odczytu i zapisu; umieszcza wskaźnik pliku na jego początku. Jeśli plik już istnieje, wywołanie <b>fopen()</b> nie powiedzie się, zwróci <b>FALSE</b> i wygeneruje błąd
	Parametr tryb otwarcia może również zawierać określenie typu pliku: b – dla plików binarnych bądź t – dla plików tekstowych.
	Parametr <b>\$użyj_include_path</b> jest to opcjonalny parametr , który może być ustawiony na '1' lub TRUE jeśli chcesz szukać pliku także w <b>include_path</b> .

## 2. Zamykanie plików

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***.

### 3. Odczyt danych.

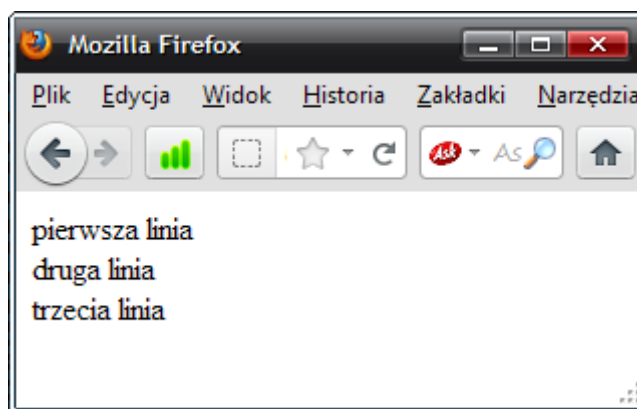
#### 3.1. Odczyt pojedynczych wierszy

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. Zobaczmy to na konkretnym przykładzie:

```
<?php
if(!$fd = fopen('./test.txt', 'r')){
    echo("Nie mogę otworzyć pliku test.txt.");
}
else{
    while(!feof($fd)){
        $str = fgets($fd);
        $str = nl2br($str);
        echo($str);
    }
    fclose($fd);
}
?>
```



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]])***

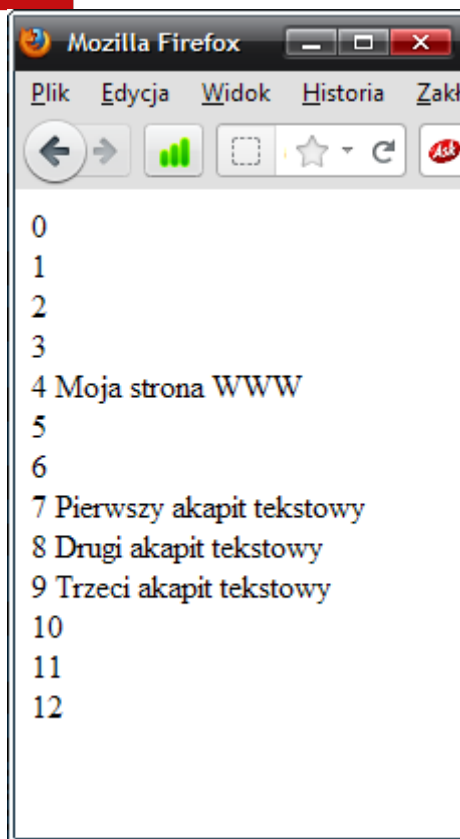
deskryptor to oczywiście wartość uprzednio zwrócona przez **fopen**, argument **ile** jest opcjonalny i określa maksymalną liczbę znaków do odczytu, a parametr **tagi** określa znaczniki, które mają nie być usuwane.

*Przykładowy plik zawierający znaczniki HTML:*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Moja strona WWW</title>
</head>
<body>
<p>Pierwszy akapit tekstowy</p>
<p>Drugi akapit tekstowy</p>
<p>Trzeci akapit tekstowy</p>
</body>
</html>
```

*Odczytanie zawartości pliku z usunięciem znaczników HTML:*

```
<?php
if(!$fd = fopen('index.html', 'r')){
    echo ("Nie mogę otworzyć pliku index.html.");
}
else{
    $lineNo = 0;
    while(!feof($fd)){
        $str = fgetss($fd);
        $str = nl2br($str);
        echo($lineNo . " $str");
        $lineNo++;
    }
    fclose($fd);
}
?>
```



### 3.2. Odczyt pojedynczych znaków

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, **fgetc** zwraca wartość false.

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

### 3.3. Odczyt bloków danych

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.

Odczyt danych za pomocą funkcji **fread**

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

Alternatywna metoda odczytu danych przy użyciu funkcji **fread**

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

### 3.4. Odczyt całej zawartości pliku

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 mogę otworzyć pliku test.txt.";
}
else{
    $rozmiar = filesize("test.txt");
    echo fread($fd, $rozmiar);
    fclose($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.

```
echo file_get_contents('test.txt');
```

Oprócz opisanych do tej pory funkcji istnieje jeszcze jedna o nazwie **file**, która odczytuje zawartość pliku o zadanej nazwie i zwraca jego zawartość w postaci takiej tablicy, gdzie każda jej kolejna komórka zawiera wiersz odczytanego tekstu.

```
<?php
$arr = file('test.txt');
foreach($arr as $key => $val){
    $str = nl2br("$key. $val");
    echo $str;
}
?>
```

#### 4. Zapis danych

##### 4.1. Przydatne funkcje

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])**

gdzie:

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

```
<?php
$str = "Przykładowy wiersz tekstu.";
if(!$fd = fopen("test.txt", 'wb')){
    echo "Nie mogę otworzyć pliku test.txt.";
}
else{
    if(fwrite($fd, $str) === false){
        echo "Wystąpił błąd. Zapis nie został dokonany.";
    }
    else{
        echo "Ciąg został zapisany.";
    }
    fclose($fd);
}
?>
```

Drugą funkcją, która pozwala na zapis danych do pliku, jest ***file\_put\_contents***. Ta funkcja może przyjmować do czterech argumentów. Schematyczne wywołanie jest następujące:

***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;

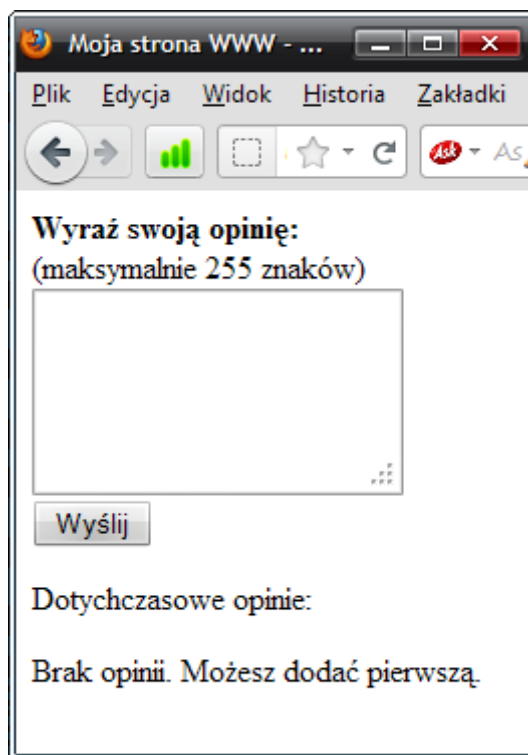
- FILE\_USE\_INCLUDE\_PATH – obecność pliku będzie sprawdzana w lokalizacjach wskazanych przez zmienną konfiguracyjną include\_path.
- FILE\_APPEND – dane będą dopisywane na końcu pliku
- LOCK\_EX – plik zostanie zablokowany na czas zapisu.

Dopisywanie danych na końcu pliku

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

#### 4.2. Zapis danych w praktyce

Wykorzystajmy teraz zdobyte dotychczas wiadomości do napisania kodu, który będzie zapisywał w pliku tekstowym opinie użytkowników o witrynie WWW. Na stronie będzie wyświetlany formularz umożliwiający wprowadzenie dowolnego tekstu oraz dotychczas wprowadzone opinie. Opinie wprowadzone przez użytkowników będą zapisywane w pliku o nazwie opinie.txt.



Skrypt będzie wykonywał trzy następujące zadania:

- Dopisanie opinii na końcu pliku
- Wyświetlenie formularza
- Wyświetlenie dotychczasowych opinii z pliku

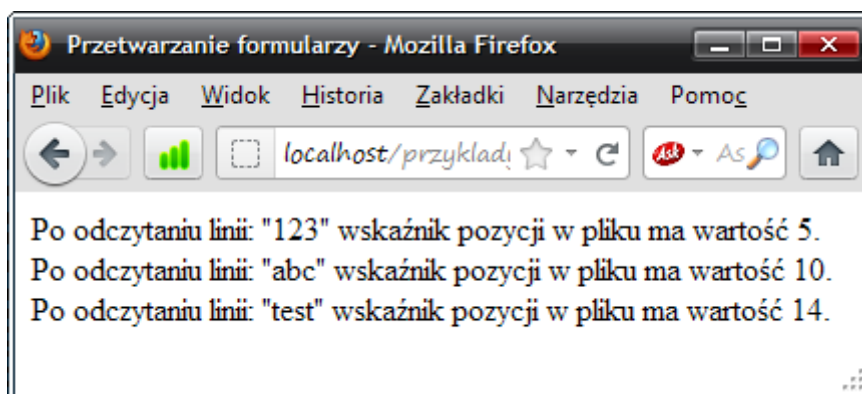


## 5. Inne operacje

## 5.1. Zmiana pozycji wewnątrz pliku

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\" ");
        echo("wskaźnik pozycji w pliku ma wartość $pos.<br \>");
    }
    fclose($fd);
}
?>
```



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

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.

## 5.2. Synchronizacja dostępu do plików

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 wartości:

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

## Ćwiczenia do samodzielnego wykonania:

### Zadanie 1.

Napisz skrypt odwracający kolejność wierszy w pliku tekstowym (tzn. ostatni wiersz ma się stać pierwszym, przedostatni drugim itd..).

### Zadanie 2.

Napisz skrypt ukazujący liczbę odwiedzin witryny. Dane powinny być zapisywane w postaci tekstowej w pliku licznik.txt. Każde wywołanie skryptu będzie powodowało otwarcie tego pliku, odczyt znajdujących się w nim danych, zwiększenie odczytanej wartości o jeden i ponowny zapis – zaktualizowanych już danych – do pliku.

### Zadanie 3.

Napisz skrypt tworzący listę odnośników. Wszystkie adresy wraz z ich opisami przechowywane będą w pliku tekstowym. Każdy wiersz pliku będzie miał schematyczną postać:

http://adres odnośnika/ opis adresu

### Zadanie 4.

Napisz skrypt, w którym użytkownikom łączącym z wybranych adresów IP zapisanych w pliku tekstowym będzie wyświetlana inna strona niż wszystkim pozostałym.