przydatne strony

http://www.gajdaw.pl/php/pliki-w-php-cz3/print.html

http://php.net/manual/pl/book.array.php

http://www.programuj.com/artykuly/www/plikiphp.php

http://pl.wikibooks.org/wiki/PHP

http://phpkurs.pl/operacje-na-plikach/

http://home.agh.edu.pl/~horzyk/lectures/ahdydwww.php#php

Komentarze w PHP

```
//      komentarz jednolinijkowy

#      komentarz jednolinijkowy uniksowy

/*
        Komentarz wielolinijkowy
*/
```

Czy plik istnieje

W języku PHP istnieją funkcje, umożliwiające przeprowadzenie różnych operacji na plikach. Umożliwiają one odczytywanie informacji o strukturze plików i katalogów oraz odczytywanie i zapisywanie danych do plików.

Do ustalenia, czy plik lub katalog istnieje, służy funkcja

```
file_exists('nazwa');
```

Funkcja

```
boolean is_file('nazwa');
```

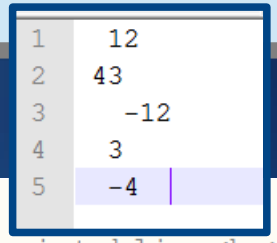
sprawdza, czy podany jako argument ciąg jest plikiem.

Klasyfikacja funkcji odczytujących dane z plików- odczytywanie całego pliku pojedynczym wywołaniem

Język PHP oferuje wiele funkcji, które służą do odczytywania zawartości plików. Funkcje te możemy scharakteryzować na dwa sposoby.

a) Funkcja file()
odczytuje podany plik, po czym w odczytanym tekście oddziela wiersze. Wynikiem działania funkcji jest tablica, której elementami są kolejne wiersze pliku. Zwróćmy uwagę na fakt, że znaki złamania wiersza nie są usuwane przez funkcje file(). Każdy z napisów zawartych w zwracanej tablicy, z wyjątkiem być może ostatniego elementu (zależy to od zawartości pliku), jest zakończony znakiem złamania wiersza. Do usunięcia znaków złamania wiersza z końca napisu możemy użyć funkcji rtrim() lub ewentualnie, w zależności od zawartości pliku- trim(). Nagłówek funkcji ograniczony do obowiązkowego parametru:

array file(string filename);



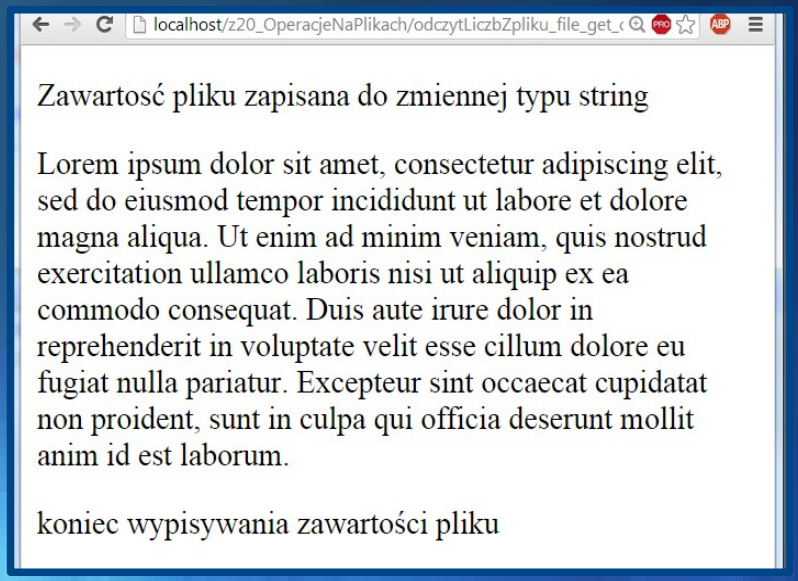

```php
<?php
echo 'Wiersze z pliku są elementami tablicy<br>';
if(file_exists('liczby.txt')){
$tab=file('liczby.txt');
 //wypiszemy elementy tablicy
 $n=count($tab);
 echo 'ilosc elementow tablicy ='.$n.'<br>';
    for($i=0;$i<$n;$i++){
     echo $tab[$i].'<br>';
    }
echo 'sprawdzimy, czy mozemy liczby dodać  <br>';
$s=0;
 for($i=0;$i<$n;$i++){
     $s+= $tab[$i];
 }
echo 'suma='.$s;
}
 else{ echo 'Plik nie istnieje';
     }
?>
```

```
Wiersze z pliku są elementami tablicy
ilosc elementow tablicy =5
12
43
-12
3
-4
sprawdzimy, czy mozemy liczby dodać
suma=42
```

Klasyfikacja funkcji odczytujących dane z plików- odczytywanie całego pliku pojedynczym wywołaniem

b) Funkcja file_get_contents(). Nagłówek funkcji ograniczony do obowiązkowego parametru:
string file_get_contents(string filename);

Funkcja ta odczytuje cały plik, którego nazwa jest pierwszym parametrem. Odczytana zawartość jest zwracana w postaci napisu. Jeśli pojawią się błędy, wówczas wartością funkcji jest false.

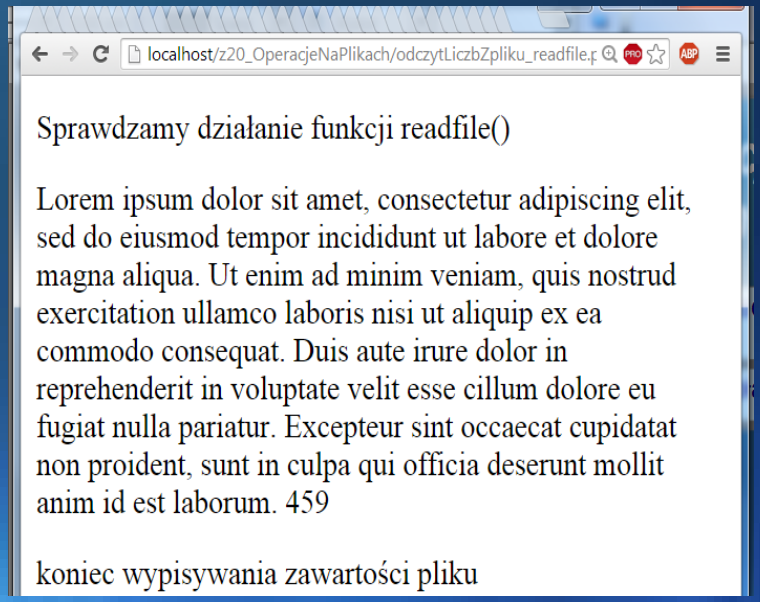



```php
<?php
echo '<p>Zawartosć pliku zapisana do zmiennej typu string</p>';
if(file_exists('plikTekstowy.txt')){
$str=file_get_contents('plikTekstowy.txt');
 echo $str;
 echo '<p>koniec wypisywania zawartości pliku</p>';
 }
 else {echo 'plik nie istnieje';
 }
?>
```

Klasyfikacja funkcji odczytujących dane z plików- odczytywanie całego pliku pojedynczym wywołaniem

c) Funkcja readfile(). Nagłówek funkcji ograniczony do obowiązkowego parametru:
int readfile(string filename);

Funkcja odczytuje zadany plik, po czym wysyła odczytaną zawartość do strumienia wyjściowego (czyli do przeglądarki, która odwiedziła skrypt). W przypadku powodzenia, wartością funkcji jest ilość przesłanych bajtów, zaś w przypadku błędu - wartość false.




```
<body>
<?php
echo '<p>Sprawdzamy działanie funkcji readfile()</p>';

 if (file_exists('plikTekstowy.txt')){
   $ilosc=readfile('plikTekstowy.txt');
   echo $ilosc;
   echo '<p>koniec wypisywania zawartości pliku</p>';
 }
 else {
     echo 'Plik  nie istnieje';
  }
?>
</body>
```

Klasyfikacja funkcji odczytujących dane z plików

	Odczyt całego pliku	Odczyt fragmentu pliku
Odczyt bez przetwarzania	`readfile()`, `file_get_contents()`	`fread()`, `fgets()`, `fgetc()`
Odczyt z przetwarzaniem	`file()`,	`fgetss()`, `fscanf()`, `fgetcsv()`

Odczytywanie fragmentów pliku- etap I – funkcja open

Jeśli odczytywanie całego pliku nie jest konieczne, wówczas mamy do dyspozycji funkcje fread(), fgets(), fgetc(), fgetss(), fgetcsv() oraz fscanf().Pierwszym parametrem każdej z nich jest uchwyt pliku. Jeśli podczas odczytywania danych wystąpią błędy, wówczas każda z podanych funkcji zwraca wartość false.

1. Otwarcie pliku funkcją fopen(). Wartość zwracaną przez funkcję fopen() nazywamy deskryptorem pliku (zmienną plikową). Deskryptor pliku jest przekazywany jako jeden z argumentów do funkcji odczytujących oraz zapisujących dane. Wszystkie funkcje obsługi plików (oprócz tej otwierającej plik) jako parametr pobierają tzw. „wskaźnik do pliku”- uchwyt pliku (ang. „file pointer”). Jest to wartość zmiennej określająca otwarty plik. Jest to niezbędne, ponieważ skrypt może otworzyć jednocześnie wiele plików i na wszystkich jednocześnie pracować.

```
$uchwytPliku = fopen("nazwaPlikuLubSciezka","tryb");
```

Wewnętrzny wskaźnik pliku

Przykład fseek()

```php
<?php

$fp = fopen('somefile.txt');

// read some data
$data = fgets($fp, 4096);

// move back to the beginning of the file
// same as rewind($fp);
fseek($fp, 0);

?>
```

Wewnętrzny wskaźnik pliku oznacza miejsce w pliku oznaczające ‚bieżącą lokalizację' – czyli miejsce w którym zaczęłoby się pisanie lub czytanie. Większość funkcji, które operują na plikach powodują przesunięcie tego wskaźnika – każde zapisanie i odczyt z pliku powodują przesunięcie tego wskaźnika. Można jednak wymusić takie przesunięcie – na przykład żeby zacząć czytanie od pewnego miejsca lub żeby pominąć jakiś fragment danych. Służy do tego funkcja

fseek(int wskaźnik_pliku, int przesunięcie [, int typ_przesunięcia]) .

Funkcja pobiera dwa obowiązkowe argument i jeden opcjonalny. Pierwszy parametr to wskaźnik pliku. Drugi to numer znaku na który ma być przesunięty wskaźnik. Ostatni, opcjonalny parametr określa typ przesunięcia (możliwe wartości to:
SEEK_SET,
SEEK_CUR,
SEEK_END).
Domyślną wartością jest SEEK_SET. Oznacza to, że wewnętrzny wskaźnik pliku zostanie ustawiony na pozycji przesunięcie.
Ustawienie ostatniego parametru na SEEK_CUR spowoduje, że wskaźnik pliku zostanie przesunięty o 'przesunięcie' znaków od bieżącego miejsca.
Natomiast wybranie typu SEEK_END spowoduje, że wskaźnik zostanie ustawiony o' przesunięcie' znaków za końcem pliku.
Funkcja
rewind(int wskaźnik_plik),
która przesuwa wewnętrzny wskaźnik pliku na początek pliku.

Funkcja open-tryb otwarcia pliku

Tryb otwarcia pliku ogranicza zbiór operacji, jakie możemy wywoływać w odniesieniu do danego pliku.

lp.	Tryb	Dopuszczalne operacje	Położenie wskaźnika odczytu/zapisu	Uwagi
1.	r	Wyłącznie odczyt.	Na początku pliku.	-
2.	r+	Odczyt/zapis.	Na początku pliku.	-
3.	w	Wyłącznie zapis.	Na początku pliku.	Jeśli plik istniał i był niepusty, to jego zawartość zostaje usunięta (tj. plik będzie miał długość 0 B). Jeśli plik nie istniał to zostanie utworzony.
4.	w+	Odczyt/zapis.	Na początku pliku.	Jeśli plik istniał i był niepusty, to jego zawartość zostaje usunięta (tj. plik będzie miał długość 0 B). Jeśli plik nie istniał to zostanie utworzony.
5.	a	Wyłącznie zapis.	Na końcu pliku.	plik tylko do zapisu; wewnętrzny wskaźnik pliku umieszczany jest na końcu pliku; jeśli plik nie istnieje PHP próbuje go stworzyć
6.	a+	Odczyt/zapis.	Na końcu pliku.	Jeśli plik nie istniał to zostanie utworzony.
7.	x	Wyłącznie zapis.	Na początku pliku.	Tworzy nowy plik. Jeśli plik istnieje, to funkcjafopen() zwróci wartośćfalse. Jeśli plik nie istniał to zostanie utworzony. Tryb dostępny wyłącznie dla plików lokalnych.
8.	x+	Odczyt/zapis.	Na początku pliku.	Tworzy nowy plik. Jeśli plik istnieje, to funkcjafopen() zwróci wartośćfalse. Jeśli plik nie istniał to zostanie utworzony. Tryb dostępny wyłącznie dla plików lokalnych.

Funkcja open-tryb otwarcia pliku-wynik działania funkcji

Wykonując operacje dostępu do plików należy mieć na uwadze ewentualne błędy. Błędy takie mogą pojawić się z wielu powodów. Typowymi przykładami są brak pliku, który chcemy otworzyć w trybie do odczytu lub niewystarczające uprawnienia do otworzenia pliku w trybie do zapisu. Błędy wykryjemy sprawdzając wyniki zwracane przez funkcje.

```php
<?php
$plik = fopen("ala.txt",'r');
if ($plik === false) {
    echo "Błąd otwarcia pliku!";
} else {
   echo "Plik otwarty";
   }
?>
```

Warning: fopen(ala.txt): failed to open stream: No such file or directory in **D:\E14\StronyInternetowe\z20_OperacjeNaPlikach\obslugaBledowPrzyOtwarciuPliku.php** on line **10**
Błąd otwarcia pliku!

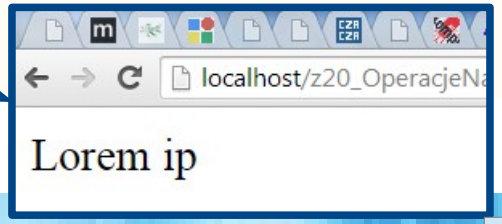
Odczyt fragmentu pliku- etap II

a) Funkcja fread(), o nagłówku:

string fread(resource handle, int length)

odczytuje z pliku co najwyżej length bajtów. Czytanie kończy się w przypadku napotkania znaku końca pliku lub odczytania podanej liczby bajtów.

```
<body>
<?php
//funkcja open zwraca uchwyt pliku
$uchwytPliku = fopen("plikTekstowy.txt",'r');
//wykorzystując uchwyt pliku, odczytujemy fragment pliku (8 znaków)
$str=fread($uchwytPliku,8);
echo $str;
?>
</body>
```



meout_przyklad_galeria.html | odczytLiczbZpliku_readfile.php | plikTekstowy.txt | odczytLiczbZpliku_file_get_contents.php

```
Lorem ipsum dolor sit amet, consectetur
adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore
magna aliqua. Ut enim
ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut
aliquip ex ea commodo consequat. Duis aute irure dolor in
reprehenderit in
voluptate velit esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui
```

```
<?php
//funkcja open zwraca uchwyt pliku
$uchwytPliku = fopen("plikTekstowy.txt",'r');
//filesize zwróci wielkość pliku;
$str=fread($uchwytPliku,filesize("plikTekstowy.txt"));
echo $str;
?>
```

int filesize(string nazwa_pliku)- funkcja zwraca wielkość pliku. Funkcja pobiera jako parametr nazwę pliku a nie uchwyt

Funkcja fread() będzie czytać dane bez przerwy – od początku pliku do końca, ignorując znaki końca linii – dla tej funkcji to zwykły znak.

Odczyt fragmentu pliku- etap II

b) Funkcja fgets() , o nagłówku:

```
bool feof ( resource $uchwyt )
```

Sprawdza czy wskaźnik pliku jest na końcu pliku (EOF).

string fgets(resource handle [, int length])
różni się od funkcji fread() tym, że czytanie jest przerywane również po napotkaniu znaku złamania wiersza. Innymi słowy odczytywane są kolejne linie pliku. Maksymalna liczba odczytywanych bajtów jest parametrem opcjonalnym, którego wartość domyślna wynosi 1024 B

```php
<body>
<?php
//funkcja open zwraca uchwyt pliku
$uchwytPliku = fopen("plikTekstowy.txt",'r');
//wykorzystując uchwyt pliku, odczytujemy wiersz pliku
$str1=fgets($uchwytPliku);
echo $str1.'<br>';
//wykorzystując uchwyt pliku, odczytujemy fragment pliku
$str2=fgets($uchwytPliku,5);
echo $str2.'<br>';
// a teraz przeczytamy pozostałą część pliku:
while(!feof($uchwytPliku)){
$str=fgets($uchwytPliku);
echo $str.'<br>';
}
?>
</body>
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolo
re magna aliqua. Ut enim
ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in
voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui

setTimeout_przyklad_galeria.html | odczytLiczbZpliku_readfile.php | plikTekstowy.txt | czytanieFragmentuPliku_fgets.php

```
1 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
  incididunt ut labore et
2 dolore magna aliqua. Ut enim
3 ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex
  ea commodo consequat. Duis aute irure dolor in reprehenderit in
4 voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint
  occaecat cupidatat non proident, sunt in culpa qui
```

Odczyt fragmentu pliku- etap II

c) Funkcja fgetc() , o nagłówku:

string fgetc(resource handle)

odczytuje z pliku jeden znak

```
bool feof ( resource $uchwyt )
```

Sprawdza czy wskaźnik pliku jest na końcu pliku (EOF).

```
<body>
<?php
//funkcja open zwraca uchwyt pliku
$uchwytPliku = fopen("plikTekstowy.txt",'r');
while(!feof($uchwytPliku)){
//do zmiennej $str zapisany jest jeden znak
$str=fgetc($uchwytPliku);
echo $str.'<br>';
}//petla kończy działanie gdy napotka znak końca pliku
?>
</body>
```

L o r e m i p s u m d

```
<?php
//funkcja open zwraca uchwyt pliku
$uchwytPliku = fopen("plikTekstowy.txt",'r');
for($i=1;$i<10;$i++){
//do zmiennej $str zapisany jest jeden znak
$str=fgetc($uchwytPliku);
echo $str;
}//wypisze 9 znaków z pliku
?>
```

Lorem ips

setTimeout_przyklad_galeria.html | odczytLiczbZpliku_readfile.php | plikTekstowy.txt | czytanieFragmentuPliku_fgets.php

```
1 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
  incididunt ut labore et
2 dolore magna aliqua. Ut enim
3 ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex
  ea commodo consequat. Duis aute irure dolor in reprehenderit in
4 voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint
  occaecat cupidatat non proident, sunt in culpa qui
```

Odczyt fragmentu pliku- etap II

d) Funkcja fgetss() , o nagłówku:

string fgetss(resource handle, int length [, string allowable_tags])

Jej działanie jest niemal identyczne jak działanie funkcji fgets(). Różnica polega na tym, że funkcja fgetss() po odczytaniu danych z pliku wykonuje automatycznie operację usuwania znaczników języka HTML. Ostatni parametr funkcji zawiera listę znaczników, które mają nie być usuwane.

Operacje na pliku- etap III - funkcja fclose()

bool fclose(resource handle)

Jej jedynym parametrem jest uchwyt pliku. Jeżeli zapomnimy o wywołaniu funkcji fclose(), to wszystkie otwarte przez skrypt pliki zostaną zamknięte przy zakończeniu działania skryptu.

Wartość funkcji fclose() mówi o powodzeniu operacji zamykania pliku.

Jeśli otworzyliśmy ten plik, to teraz trzeba go jakoś zamknąć. Jeśli nie zamkniemy pliku ręcznie, to PHP zrobi to za nas po zakończeniu działania skryptu. Nie sprawia to różnicy jeśli na stronie operujemy tylko jednym plikiem lub np. po zapisaniu danych do pliku już się nim nie interesujemy. Inaczej sprawa wygląda jeśli po zapisaniu danych do pliku chcemy później go odczytać – dopiero po zamknięciu pliku zmiany w nim będą widoczne. A do zamknięcia pliku służy funkcja fclose(int wskaźnik).

Zapis do pliku

a) Funkcja fputs() , o nagłówku:

fputs(resource handle, string napis, int długość). // ta funkcja to alias funkcji fwrite

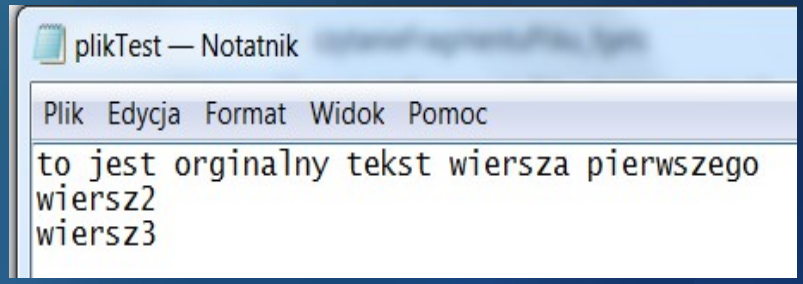
Dane do pliku można zapisać przy pomocy funkcji fputs(int wskaznik, string napis, int długość). Jak przy większości funkcji operujących na plikach, niezbędne jest podanie wskaźnika pliku na którym chcemy operować. Zapisać można albo całą zawartość zmiennej podanej jako drugi parametr, albo tylko do pewnej długości, którą należy podać jako trzeci, opcjonalny parametr (oczywiście przy pominięciu tego parametru zapisywana jest cała zmienna podana w drugim parametrze). Zapis odbywa się w miejscu, na który wskazuje wewnętrzny wskaźnik pliku, nadpisując dane jeśli wskaźnik ten nie znajduje się na końcu pliku. Nie ma żadnej możliwości bezpośredniego zapisania danych na początku lub w środku pliku. Jedyna możliwość to wczytanie pliku do tymczasowej zmiennej, poprawienie tych danych i ponowny zapis tego pliku.

```php
<?php
//funkcja open zwraca uchwyt pliku
//jeżeli nie istnieje plik o tej nazwie, funkcja utworzy plik
$uchwytPliku = fopen("plikTest.txt",'w');

fputs($uchwytPliku, "to jest tekst do zapisania w pliku");
fclose($uchwytPliku);

?>
```

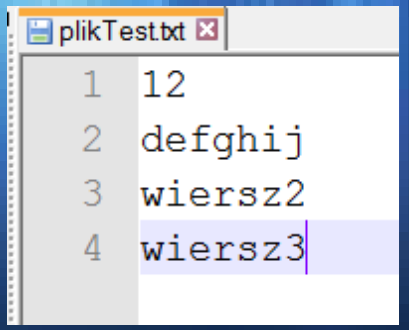
Zapis do pliku-dane nadpisane na początku pliku




```php
<?php
//funkcja open zwraca uchwyt pliku
//jeżeli nie istnieje plik o tej nazwie, funkcja utworzy plik
$uchwytPliku = fopen("plikTest.txt",'r+');
//$str=fread($uchwytPliku, filesize("plikTest.txt") );
//poniższa instrukcja nadpisze pierwszy wiersz w pliku tekstowym
//i wskaznik wewnętrzny pliku ustawi na początek wiersza 2.
$str="12\n";
fputs($uchwytPliku, $str);
fclose($uchwytPliku);

?>
```

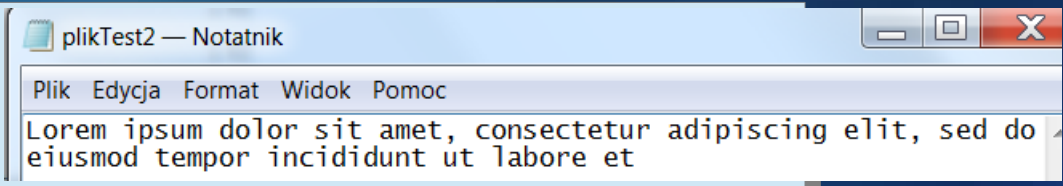
Wynik działania skryptu



Zapis do pliku-stare dane zachowane; nowe dane dopisane na początku pliku

Funkcja fputs() , o nagłówku:

1) odczytamy dane z pliku
2) zapiszemy zawartość pliku do zmiennej $str
3) na początku zmiennej $str dopiszemy nowe dane
4) zawartość zmiennej zapiszemy do pliku




```php
<?php
//funkcja open zwraca uchwyt pliku
$uchwytPliku = fopen("plikTest2.txt",'r+');
$str=fread($uchwytPliku, filesize("plikTest2.txt") );
//zawartość pliku zapisana do zmiennej
//wewnętrznywskaźnik pliku ustawiony na końcu pliku
rewind($uchwytPliku);
// wewnętrzny wskaznik pliku ustawiony na początku pliku
//nowe dane przed starymi danymi
$str="nowe dane \n".$str;
fputs($uchwytPliku, $str);
fclose($uchwytPliku);
?>
```

```
nowe dane
Lorem ipsum dolor sit amet,  adipiscing elit, sed do eiusmod tempor incididunt
ut labore et
```

Zapis na koniec pliku

Aby zapisać dane na koniec pliku wystarczy otworzyć plik w trybie „a” i od razu można dodawać dane do pliku.

unkcja_strlen.php | zapisNaKoniecPliku_fputs.php | plikTest.txt | plikTest2.txt | plikTekstowy3.txt

```
1 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
  eiusmod tempor incididunt ut labore et
2 dolore magna aliqua. Ut enim
3
```

```php
<?php
//funkcja open zwraca uchwyt pliku
$uchwytPliku = fopen("plikTekstowy3.txt",'a');
$str=fread($uchwytPliku, filesize("plikTekstowy3.txt") );
$str="nowe dane dodane na koniec pliku\n";
fputs($uchwytPliku, $str);
fclose($uchwytPliku);
?>
```

```
1 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt
  ut labore et
2 dolore magna aliqua. Ut enim
3 nowe dane dodane na koniec pliku
```

Zapis do pliku

b) Funkcja fwrite() , o nagłówku:

int fwrite(resource handle, string string [, int length])

Funkcja fwrite() posiada trzy parametry: deskryptor pliku handle, zapisywany napis string oraz opcjonalny parametr length ograniczający ilość zapisanych bajtów. Jeśli trzeci parametr nie jest podany, to do pliku zostaje zapisany cały podany napis. W przeciwnym razie, do pliku zostaje zapisanych co najwyżej length bajtów. (Być może mniej, jeżeli napis jest krótszy). Wynikiem funkcji jest ilość zapisanych bajtów.

Zarówno fread() jak i fwrite() może zwrócić - w przypadku niepowodzenia - wartość logiczną false.

```
<!DOCTYPE html>
<html lang="pl">
  <head>
    <title>działanie funkcji fwrite()</title>
    <meta charset="UTF-8" />
</head>
<body>
<?php
$uchwytPliku = fopen("plikFwrite.txt",'r+');
$str="nowe dane";
fwrite($uchwytPliku, $str);
fclose($uchwytPliku);
?>
</body>
</html>
```

Blokowanie plików

Blokowanie plików jest jednym z ważniejszych zagadanień przy używaniu plików do przechowywania danych. W zastosowaniach internetowych może dojść do takiej sytuacji, że dwie lub więcej osób jednocześnie wejdzie na stronę czy po prostu uruchomi skrypt. Jeśli będzie to na przykład licznik odwiedzin przechowujący ilość odwiedzin w pliku, to te kilka osób będzie chciało zapisać dane do tego pliku jednocześnie. Może to doprowadzić do utraty danych z tego licznika. Wystarczy jednak zastosować mechanizm blokad aby zapobiec takiej sytuacji.

Funkcja flock(resource wskaźnik_do_pliku, int operacja)

zakłada blokadę lub ją zdejmuje, zależnie od wartości drugiego parametru. Są 2 typy blokad: blokada dzielona, używana jeśli plik ma być odczytywany (dzielona, ponieważ więcej niż jeden skrypt może utrzymywać taką blokadę na pliku) , i blokada wyłączna, zakładana jeśli plik ma być zapisywany.

Przycinanie plików

PHP zawiera funkcję służącą do zmniejszania rozmiaru pliku do zadanej wielkości.
Jest nią funkcja:

bool ftruncate (resource $uchwyt , int $rozmiar)

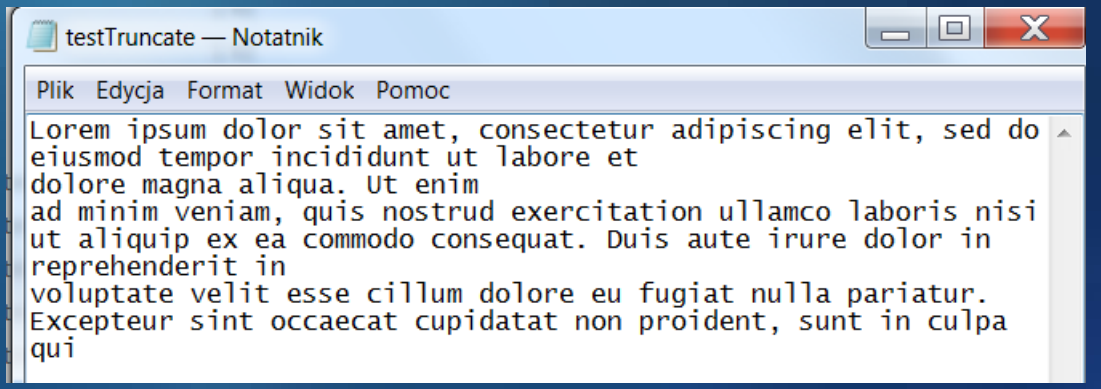
Pobiera wskaźnik pliku i przycina plik do długości rozmiar .

Uwaga: plik musi by otworzony do zapisu.

Uwaga: Jeśli rozmiar jest większy niż rozmiar pliku to jest on rozszerzany bajtami null.
Jeśli rozmiar jest mniejszy niż rozmiar pliku to dalsze dane zostaną utracone.

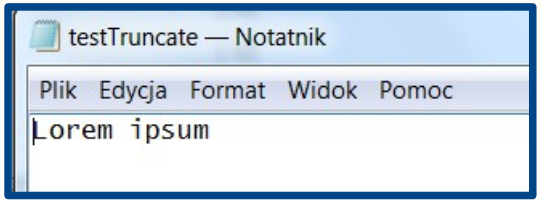
Zwracane wartości:
Zwraca TRUE w przypadku powodzenia, FALSE w przypadku błędu.

Przycinanie plików




```php
<!DOCTYPE html>
<html lang="pl">
  <head>
    <title>działanie funkcji truncate()</title>
    <meta charset="UTF-8" />
</head>
<body>
<?php
$uchwytPliku = fopen("testTruncate.txt",'r+');
if(ftruncate($uchwytPliku,12)){echo "plik został skrócony ";}
   else {echo "operacja nie powiodła się ";}

fclose($uchwytPliku);
?>
</body>
</html>
```



Blokowanie plików

Blokowanie plików jest jednym z ważniejszych zagadanień przy używaniu plików do przechowywania danych. W zastosowaniach internetowych może dojść do takiej sytuacji, że dwie lub więcej osób jednocześnie wejdzie na stronę czy po prostu uruchomi skrypt. Jeśli będzie to na przykład licznik odwiedzin przechowujący ilość odwiedzin w pliku, to te kilka osób będzie chciało zapisać dane do tego pliku jednocześnie. Może to doprowadzić do utraty danych z tego licznika.
Funkcja flock(int wskaźnik, int operacja) zakłada blokadę lub ją zdejmuje, zależnie od wartości drugiego parametru. Są 2 typy blokad: blokada dzielona,
używana jeśli plik ma być odczytywany (dzielona, ponieważ więcej niż jeden skrypt może utrzymywać taką blokadę na pliku) , i blokada wyłączna, zakładana jeśli plik ma być zapisywany.

```
10 $file = fopen("test.txt","w+");
11 // exclusive lock
12 if (flock($file,LOCK_EX))
13   {
14   fwrite($file,"Write something");
15   // release lock
16   flock($file,LOCK_UN);
17   }
18 else
19   {
20   echo "Error locking file!";
21   }
22
23 fclose($file);
```

Drugi parametr funkcji może mieć następujące wartości:

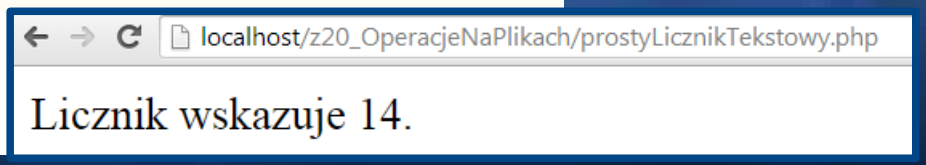
LOCK_SH (-nazwa stałej, której przypisano wartość 1) aby założyć blokadę dzieloną
LOCK_EX (-nazwa stałej, której przypisano wartość 2) aby założyć blokadę wyłączną
LOCK_UN (-nazwa stałej, której przypisano wartość 3) aby zdjąć blokadę, wszystko jedno jaką.

Otóż funkcja flock() zwraca wartość true lub false w zależności od tego, czy udało się założyć blokadę czy nie. Przykładowo, jeśli na pliku założona jest blokada dzielona, to można ten plik jeszcze raz zablokować do odczytu, ale do zapisu już nie. Jeśli natomiast założona jest blokada wyłączna, to żaden inny skrypt nie może już założyć blokady do czasu aż ta blokada zostanie zdjęta.

Blokowanie plików

```
<head>
  <title>działanie prostego licznika tekstowego</title>
  <meta charset="UTF-8" />
</head>
<body>
<? // Program pokazuje działanie prostego licznika tekstowego.
   // każde odświeżenie strony zwiększa wartość licznika o 1
    if (!(file_exists("licznik.txt"))) { // nie ma pliku z licznikiem
      $plik = fopen ("licznik.txt", "w+");//więc go tworzymy
      fputs ($plik, "0");       //zapisujemy wartosc 0
      fclose ($plik);
    }

    $plik = fopen ("licznik.txt", "r+");
    if (!($plik)) {
      print "Błąd otwarcia pliku: Nie da się otworzyć pliku.";
    } else {
      flock ($plik, 2);//blokujemy plik - mozna tak:   flock ($plik,  LOCK_EX)
      $ile = fgets ($plik, 20);
      print "Licznik wskazuje $ile.";
      $ile++;
      fseek ($plik, 0);//wewnetrzny wskaznik pliku ustawiamy na poczatek pliku
      fputs ($plik, "$ile");//zapisujemy liczbe
      flock ($plik, 3);//odblokowujemy plik-mozna tak:  flock ($plik, LOCK_UN);
      fclose ($plik);
    }
  ?>
</body>
</html>
```



Funkcje logiczne (zwracają wartość true lub false)

file_exists($nazwa_pliku) - sprawdza, czy plik o podanej nazwie istnieje

is_dir($nazwa_pliku)	czy plik o podanej ścieżce jest katalogiem
is_executable($nazwa_pliku)	czy plik jest wykonywalny
is_file($nazwa_pliku)	czy plik jest normalnym plikiem
is_link($nazwa_pliku)	czy plik jest dowiązaniem
is_readable($nazwa_pliku)	czy plik można czytać
is_writable($nazwa_pliku)	czy do pliku można pisać
is_uploaded_file($nazwa_pliku)	czy plik został przesłany z formularza
is_executable($nazwa_pliku)	czy plik jest wykonywalny
is_file($nazwa_pliku)	czy jest plikiem

Prawa dostępu (tylko UNIX)

Unix i pochodne, jako systemu przeznaczone dla wielu użytkowników, zawierają system zabezpieczania dostępu do plików przez niepowołane osoby. System ten opiera się o przydzielanie praw poszczególnym osobom (a konkretniej ich kontom systemowym) oraz grupom, do których użytkownicy należą.

Każdemu plikowi i katalogowi w systemie można przypisać 3 komplety praw. Każdy z tych kompletów to:

- prawo wykonywania (litera ‚x' lub liczba 1) – w przypadku katalogów oznacza to możliwość wejścia do katalogu,
- prawo zapisu (litera ‚w' lub liczba 2),
- prawo odczytu (litera ‚r' lub liczba 4).

Pierwszy komplet tych praw dotyczy właściciela pliku, drugi- grupy, a trzeci użytkowników, którzy nie są właścicielami ani nie należą do grupy. Liczby przypisane konkretnym prawom służą do zapisu ósemkowego. Zapis taki to 3 cyfry, z których każda odpowiada kompletowi praw – sumie liczb oznaczających prawa.

- Przykładowo, liczba 7 to komplet praw wykonywania, zapisu i odczytu (1+2+4=7) a liczba 5 to prawo odczytu i wykonywania (1+4=5). Czyli aby nadać właścicielowi komplet praw, grupie odczyt i zapis a reszcie tylko odczyt, należy ustawić prawa „0764" (cyfra zero na początku służy do poinformowania PHP, że liczba zapisana jest w formacie ósemkowym).

Prawa dostępu (tylko UNIX)

Do ustawiania praw dostępu służy funkcja chmod($nazwa_pliku, $tryb), gdzie drugi parametr to prawa dostępu zapisane w formacie ósemkowym. Inne funkcje przydatne przy pracy z systemem zabezpieczeń to chown($nazwa_pliku, $user), zmieniająca właściciela pliku, i chgrp($nazwa_pliku, $grupa), zmieniająca grupę.

Ustawienie odpowiednich praw dostępu to sprawa bardzo ważna w przypadku skryptów zapisujących coś do plików. Błąd „Permission denied” zdarza się bardzo często i można go rozwiązać właśnie przez ustawienie odpowiednich praw. Aby skrypt mógł coś zapisać w pliku, niezbędne jest ustawienie prawa zapisu do tego pliku.

Uwaga - zmiana praw nie działa na plikach zdalnych lub gdy włączony jest tryb bezpieczny – safe_mode.

Przykład:

```
$nazwaPliku="licznik2.txt";
 chmod($nazwaPliku, 0777);
```

Inne przykłady przyznanych liczbowo praw:
0600 - Odczyt i zapis dla właściciela, żadnych praw dla innych
0644 - Odczyt i zapis dla właściciela, odczyt dla wszystkich
0755 - Wszystkie prawa dla właściciela, odczyt i wykonanie dla innych
0750 - Wszystkie prawa dla właściciela, odczyt i wykonanie dla grupy właściciela.

Funkcje informacyjne

Przy skryptach przeznaczonych do obsługi serwera czy na przykład analizujących system plików niezbędne jest uzyskanie informacji o konkretnym pliku. PHP oferuje cały zestaw funkcji zwracających dane o pliku o podanej ścieżce.

fileatime($nazwa_pliku)	zwraca czas ostatniego odczytu pliku; czas ten jest zwracany w postaci timestamp
filectime($nazwa_pliku)	zwraca czas ostatniej modyfikacji i-węzła (dotyczy tylko systemów Unix) w formacie timestamp
filemtime($nazwa_pliku)	zwraca czas ostatniej modyfikacji pliku w formacie timestamp
fileowner($nazwa_pliku)	zwraca identyfikator użytkownika, który jest właścicielem pliku
filegroup($nazwa_pliku)	zwraca identyfikator grupy, do której należy plik (tylko Unix)
fileinode($nazwa_pliku)	zwraca numer i-węzła do którego przypisany jest plik (tylko Unix)
fileperms($nazwa_pliku)	zwraca prawa dostępu do pliku
filesize($nazwa_pliku)	zwraca wielkość pliku w bajtach
filetype($nazwa_pliku)	zwraca typ pliku (tylko UNIX); możliwe typy to „fifo”, „char”, „dir”, „block”, „link”, „file”, „unknown” dla odpowiednio kolejek fifo, urządzeń znakowych, katalogów, urządzeń blokowych, dowiązań, zwykłych plików i nieznanych typów
stat($nazwa_pliku)	funkcja ta zwraca tablicę zawierającą pełne informacje o pliku.

Funkcje informacyjne-przykład

```php
<? // Program pokazuje działanie prostego licznika tekstowego.
   // każde odświeżenie strony zwiększa wartość licznika o 1
  $nazwaPliku="licznik2.txt";

    if (!(file_exists( $nazwaPliku))) { // nie ma pliku z licznikiem
      $plik = fopen ($nazwaPliku, "w+");//więc go tworzymy
      fputs ($plik, "0");       //zapisujemy wartosc 0
      fclose ($plik);
    }
    $plik = fopen ( $nazwaPliku, "r+");
    if (!($plik)) {
      print "Błąd otwarcia pliku: Nie da się otworzyć pliku.";
    } else {
      flock ($plik, 2);//blokujemy plik - mozna tak:   flock ($plik,  LOCK_EX)

      $ile = fgets ($plik, 20);
      print "Licznik wskazuje $ile.";
      $ile++;
      fseek ($plik, 0);//wewnetrzny wskaznik pliku ustawiamy na poczatek pliku
      fputs ($plik, "$ile");//zapisujemy liczbę
      flock ($plik, 3);//odblokowujemy plik-mozna tak:  flock ($plik, LOCK_UN);
      fclose ($plik);
    }
    print "<br>identyfikator użytkownika, który jest właścicielem pliku: ".fileowner($nazwaPliku);
     print "<br>prawa dostępu do pliku: ".fileperms($nazwaPliku);
       ?>
```

Licznik wskazuje 29.
identyfikator użytkownika, który jest właścicielem pliku: 1293610
prawa dostępu do pliku: 33279

Dec 33279
Oct 100777

Podsumowanie- Klasyfikacja przypadków dostępu do pliku

O dostępie do pliku mówimy wówczas, gdy skrypt wywołuje którąkolwiek z funkcji otwarcia, zamknięcia, odczytu, zapisu, zablokowania, modyfikacji położenia wskaźnika oraz uzyskania informacji o pliku. Wywoływanie tych funkcji, w zależności od okoliczności, podlega większym lub mniejszym ograniczeniom.

Wpływ na to mają dwa czynniki: tryb dostępu do pliku oraz współbieżność wykonania skryptów. Powyższe kryteria rozważamy w stosunku do każdego pliku.

Podsumowanie-
Klasyfikacja przypadków dostępu do pliku

1. - tryb dostępu do pliku.

- **odczyt pliku czytanego** - plik jest otwierany w danym skrypcie wyłącznie w trybie do odczytu. Żaden inny skrypt nie otwiera pliku w trybie do zapisu (tj. każdy skrypt korzystający z danego pliku otwiera go jedynie w trybie do odczytu). Plik taki jest tworzony przez administratora poza PHP, na przykład edytorem plików tekstowych. Zawartość pliku może służyć do przygotowania menu widocznego na stronie, listy dostępnych plików do pobrania, czy spisu treści.

- **odczyt pliku zapisywanego** - plik jest otwierany przez dany skrypt w trybie do odczytu, jednakże inne skrypty mogą otwierać plik do zapisu. Przykładem takiej sytuacji jest ponownie licznik odwiedzin. W zależności od zawartości ciasteczek, czasami wizyty nie powodują zapisu informacji w pliku. Wartość zapisana w pliku jest jedynie odczytywana w celu wyświetlenia na stronie.

- **odczyt/zapis** - plik jest otwierany przez dany skrypt w trybie do zapisu z ewentualnym odczytem danych (np. w trybie `'r+'`). Plik jest odczytywany, przetwarzany i zapisywany przez skrypt. Przykładem może być licznik odwiedzin zwiększany przy każdej wizycie.

Podsumowanie-
Klasyfikacja przypadków dostępu do pliku

2. - współbieżność wykonania operacji plikowych-wprowadza podział na skrypty:

- **administracyjne** - tj. takie, które nie są dostępne poprzez WWW. Skrypty administracyjne są wykonywane wyłącznie przez administratora systemu. Nie pojawia się wówczas problem współbieżności wykonania.

Skryptami administracyjnymi są między innymi skrypty konwertujące dokumenty tworzące witrynę (np. zmiana kodowania polskich znaków lub zmiana krótkich znaczników php `<?` w znaczniki pełne `<?php`).

W stosunku do skryptów administracyjnych przyjmiemy dwa założenia:

- na czas wykonywania skryptu administracyjnego witryna jest niedostępna *on-line*,
- skrypt nie jest wykonywany współbieżnie (tj. po uruchomieniu skryptu, administrator czeka na jego zakończenie przed ewentualnym ponownym uruchomieniem).

Podsumowanie- Klasyfikacja przypadków dostępu do pliku

2. - współbieżność wykonania operacji plikowych-wprowadza podział na skrypty:

- **on-line** - tj. takie, które są wykonywane podczas wizyt na naszej witrynie. Ponieważ witryna może zostać odwiedzona przez wielu internautów na raz, zatem skrypty takie muszą uwzględniać współbieżność wykonania.

Przykładowym skryptem dostępnym *on-line* jest licznik wizyt na stronie. Strona jest odwiedzana przez wielu gości, każda wizyta powoduje zapisanie informacji do pliku.

Podsumowanie- Klasyfikacja przypadków dostępu do pliku

Tabela przedstawia sumaryczne zestawienie sytuacji, jakie mogą mieć miejsce przy uwzględnieniu powyższych dwóch klasyfikacji.

	Odczyt pliku czytanego	Odczyt/zapis	Odczyt pliku zapisywanego
Administracyjne	Niekonieczna blokada. Odczyt: `file()`, `file_get_contents()`, `file_n()`, dowolne inne	Niekonieczna blokada. Odczyt: `file()`, `file_get_contents()`, `file_n()`, dowolne inne Zapis: pojedyncze wywołanie `file_put_contents()`	Zakładamy brak współbieżności wykonania. Taka sytuacja nigdy nie ma miejsca.
On-line	Niekonieczna blokada. Odczyt: `file()`, `file_get_contents()`, `file_n()`, dowolne inne. Żaden skrypt nie zapisuje danych do pliku.	Konieczna blokada `LOCK_EX` Zapis: pojedyncze wywołanie `fwrite()` Odczyt: pojedyncze wywołanie `fread()`	Zalecana blokada `LOCK_SH` Odczyt: pojedyncze wywołanie `fread()`

Własna funkcja

Zwracając uwagę na powyższe podziały możemy sformułować pierwszą regułę, która ma na celu zapewnienie poprawności wykonywania operacji plikowych.

Należy jasno ustalać reguły dostępu do plików. Na początku, skrypty dzielimy na administracyjne oraz on-line. Następnie w stosunku do każdego pliku ustalamy tryb dostępu: odczyt pliku czytanego, odczyt/zapis lub odczyt pliku zapisywanego.

Podsumowanie- Klasyfikacja przypadków dostępu do pliku

Analizując tabelę ze strony 35. zauważymy, że w istocie mamy do czynienia z czterema istotnie różnymi przypadkami:

1. **Skrypt administracyjny.**

   W skryptach administracyjnych nie musimy stosować blokady dostępu do plików. Dane odczytujemy dowolnymi funkcjami (zazwyczaj najefektywniejsze będzie użycie funkcji `file_get_contents()`). Jeśli zachodzi potrzeba, to dane możemy zapisywać do plików stosując dowolne funkcje (należy rozważyć użycie funkcji `file_put_contents()` jako prawdopodobnie najefektywniejszej). W przypadku skryptu administracyjnego rozróżnienie pod względem trybu dostępu nie odgrywa istotnej roli. Brak współbieżności wykonania eliminuje główne problemy, z jakimi mamy do czynienia w przypadku zapisywania danych do plików w skryptach PHP.

Podsumowanie- Klasyfikacja przypadków dostępu do pliku

Analizując tabelę ze strony 35. zauważymy, że w istocie mamy do czynienia z czterema istotnie różnymi przypadkami:

2. **Skrypt *on-line*, odczyt pliku czytanego.**

   Żaden ze skryptów korzystających z danego pliku nie może do niego zapisywać informacji. Nie musimy blokować dostępu do pliku. Do odczytu pliku stosujemy dowolne funkcje, w szczególności `file_get_contents()` oraz `file()`.

3. **Skrypt *on-line*, odczyt/zapis pliku.**

   Sytuacja najbardziej problematyczna. Należy uwzględnić wszystkie możliwe zabezpieczenia. Konieczne blokowanie dostępu do pliku „na wyłączność” (blokada `LOCK_EX`). Do odczytu pliku stosujemy jedno wywołanie funkcji `fread()`. Nie możemy – z racji na konieczność blokowania pliku – korzystać z funkcji `file()`, `file_get_contents()`, itd. Do zapisania zawartości pliku stosujemy pojedyncze wywołanie funkcji `fwrite()`. Nie możemy – z racji na konieczność blokowania dostępu – stosować funkcji `file_put_contents()`.

Podsumowanie- Klasyfikacja przypadków dostępu do pliku

Analizując tabelę ze strony 35. zauważymy, że w istocie mamy do czynienia z czterema istotnie różnymi przypadkami:

4. **Skrypt *on-line*, odczyt pliku pisanego.**

   Z racji na to, że inne skrypty wykonujące się współbieżnie stosują blokowanie danego pliku, zalecane jest również blokowanie dostępu. Zakładamy blokadę `LOCK_SH`, która umożliwia współbieżne wykonywanie operacji czytania danych z pliku przez wiele procesów. Do odczytu pliku stosujemy jedno wywołanie funkcji `fread()`. Nie korzystamy – z racji na zakładaną blokadę – z funkcji `file()`, `file_get_contents()`, itd. Oczywiście do pliku nie możemy zapisywać danych.

Podsumowanie-zastosowanie praktyczne-odczyt pliku „do odczytu”

Pierwszym przypadkiem, jaki rozważymy praktycznie, jest odczyt danych z pliku otwieranego wyłącznie do odczytu. Jeśli plik jest wyłącznie odczytywany przez skrypty i żaden ze skryptów nigdy nie otwiera go w trybie do zapisu, wówczas — bez względu na to czy rozważamy skrypt administracyjny czy on-line — nie musimy stosować zabezpieczeń dostępu do pliku.

W takiej sytuacji możemy korzystać z dowolnej spośród funkcji odczytujących dane z plików: `readfile()`, `file_get_contents()`, `file()`, `parse_ini_file()`, `fread()`, `fgets()`, `fgetc()`, `fgetss()`, `fscanf()` oraz `fgetcsv()`. Zwróćmy jednak uwagę na fakt, że jeśli przeczytany ma być cały plik do zmiennej będącej napisem, to zdecydowanie najlepiej użyć funkcji `file_get_contents()`. Natomiast w przypadku, gdy wymagany jest podział na wiersze stosujemy funkcję `file()`. Wreszcie w przypadku, gdy potrzebnych jest kilka początkowych wierszy pliku, należy przygotować własną funkcję `file_n()`, która odczyta wymaganą ilość linii i zwróci ją w odpowiedniej postaci. W innych, bardziej specyficznych przypadkach, stosujemy własne implementacje odczytu danych.

Podsumowanie-zastosowanie praktyczne-odczyt pliku „do odczytu”

Odczytamy zawartość pliku tekstowego, a następnie wyślemy do przeglądarki, zamieniając znaki złamania wiersza w znaczniki HTML:

```
<body>
<h4>zamiana znaków konca linii na "br" </h4>
<?php
$plik=file_get_contents('PRZEDSZKOLA.txt', false); //zawartość pliku
zapisana w zmiennej string;
//funkcja pobiera zawartość pliku wraz ze znakami \n
//znaki końca linii  \n nie zostały zamienione na znacznik <br> HTML
echo $plik;//zawartość pliku, bez znaków nowej linii, wyświetlona na
stronie www
echo '<br>';
echo nl2br($plik, false);//funkcja zamienia znaki  znaki \n na <br> w HTML
    ?>
</body>
```

nl2br($ciagZnakow, false-jezeli konwersja na
)

Id_przedszkola;Nazwa_przedszkola;Liczba_miejsc 66;Niepubliczne Przedszkole Cogito;50
14;Niepubliczne Przedszkole Krasnal;30 13;Niepubliczne Przedszkole Kraina Teczy;30
57;Przedszkole Niepubliczne Radosny Zakatek;30 52;Przedszkole Niepubliczne
Stokrotka;25 49;Przedszkole Niepubliczne Artystyczno-Jezykowe;20 20;Przedszkole
Niepubliczne im. Panienki z Okienka;20 34;Przedszkole Niepubliczne Jezykowe

Id_przedszkola;Nazwa_przedszkola;Liczba_miej
66;Niepubliczne Przedszkole Cogito;50
14;Niepubliczne Przedszkole Krasnal;30
13;Niepubliczne Przedszkole Kraina Teczy;30

XML (ang. Extensible Markup Language, w wolnym tłumaczeniu Rozszerzalny Język Znaczników) – uniwersalny język znaczników przeznaczony do reprezentowania różnych danych w strukturalizowany sposób.

**nl2br()-konwertuje ciąg znaków z pustymi znakami typu '\n', '\r\n', '\n\r' na znacznik '
' lub '
'. Przyjmuję również opcjonalny drugi parametr, TRUE jeśli ma być zwrócony znacznik z XHTML lub FALSE dla zwykłego HTML. Domyślnie ustawiony jest na TRUE.**