Podstawowe wiadomości dotyczące operacji na plikach w PHP

1. Otwieranie i zamykanie plików

Wszystkie funkcje obsługi plików (oprócz tej otwierającej plik) jako parametr pobierają tzw. "wskaźnik do 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. Wskaźnik to pliku jest zwracany przez funkcję fopen(nazwa pliku, string tryb). Drugi parametr określa tryb otwarcia pliku. Tryb trzeba dobrać odpowiednio do tego, co się chce z plikiem robić. Możliwe tryby to:

"r" - plik tylko do odczytu; wewnętrzny wskaźnik pliku umieszczany jest na początku pliku
"r+"- plik do odczytu i zapisu; wewnętrzny wskaźnik pliku umieszczany jest na początku pliku
"w"- plik tylko do zapisu; wewnętrzny wskaźnik pliku umieszczany jest na końcu pliku; zawartość pliku jest niszczona (długość pliku jest zmieniana na zero); jeśli plik nie istnieje PHP próbuje go stworzyć
"w+"- plik do odczytu i do zapisu; wewnętrzny wskaźnik pliku umieszczany jest na końcu pliku; zawartość pliku jest niszczona (długość pliku jest zmieniana na zero); jeśli plik nie istnieje
"a"- plik tylko do zapisu; wewnętrzny wskaźnik pliku umieszczany jest na końcu pliku; jeśli plik nie istnieje PHP próbuje go stworzyć

Pozycja „wewnętrznego wskaźnika pliku" określa, w którym miejscu zaczęłoby się pisać do pliku, gdyby zaczęło się pisać lub czytać z tego pliku zaraz po otwarciu pliku. Jeśli jest na końcu – to znaczy że dane dopisywane są na końcu.

Przykład.1 Otwarcie pliku do odczytu

```
<?
$plik = fopen("dane.txt", "r");
?>
```

Zmienna $plik zawiera teraz wskaźnik do pliku dane.txt. Wskaźnik ten trzeba podać przy każdej funkcji, która w jakiś sposób operuje na tym pliku.
Jeśli otworzymy ten plik, to trzeba go 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. Do zamknięcia pliku służy funkcja fclose(int wskaźnik).

Wewnętrzny wskaźnik pliku

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, int przesunięcie [, int typ_przesunięcia. Jak widać, funkcja pobiera 2 obowiązkowe argumenty i jeden opcjonalny. Pierwszy parametr to wskaźnik pliku – funkcja musi wiedzieć w którym pliku ma przesuwać wskaźnik.
Drugi to numer znaku na który ma być przesunięty wskaźnik. Ostatni określa typ przesunięcia (możliwe wartości to: SEEK_SET, SEEK_CUR i SEEK_END). Domyślną wartością jest SEEK_SET. Oznacza to, że wewnętrzny wskaźnik pliku zostanie ustawiony na pozycji przesunięcie. Ustawienie ostatniego parametru na SEEK_CURR 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. Istnieje też funkcja rewind(int wskaźnik), który przesuwa wewnętrzny wskaźnik pliku na początek pliku.

2. Odczyt z plików

Odczyt z pliku może być przeprowadzany na 2 sposoby jeden to „niskopoziomowe" czytanie danych z pliku za pomocą funkcji fread(int wskaźnik, int długość). Czyta ona dane odczytane z pliku, określonego przez utworzony wcześniej "wskaźnik" (wskaźniki są zwracane przez funkcję fopen()). Dane są czytane do długości określonej w drugim parametrze.

Jeśli danych jest mniej niż podana długość, to po prostu zwrócone zostaną wszystkie dane z pliku.
Funkcja fread() powoduje przesunięcie wewnętrznego wskaźnika pliku – do miejsca, w którym zakończyło się czytanie. Dany jest przykładowy plik dane.txt – z treścią "To jest przyklad".

Przykład.2. Odczyt danych z pliku

```
<?php
$plik = fopen("dane.txt", "r");
$t = fread($plik, 10);
?>
```

Funkcją fopen otwarty został plik do odczytu a następnie odczytanych zostało z niego 10 znaków, a więc zmienna $t będzie zawierała "To jest pr". Przesunięty został wskaźnik pliku, a więc następne wywołanie linii "$t = fread($plikp, 10)" spowoduje, że w tej zmiennej będzie tekst "zykładowy" – poprostu następne czytanie zaczęło się od miejsca, w który zakończyło czytać poprzednie wywołanie tej funkcji.
Jeszcze może praktyczny przykład – jak wczytać cały plik do zmiennej? Trzeba do tego użyć jeszcze jednej funkcji – filesize(string nazwa_pliku). Jak widać, różni się ona od pozostałych funkcji tym, że nie pobiera jako parametr wskaźnika do pliku ale nazwę pliku. Pobieranie całości pliku do zmiennej można załatwić jedną linijką.

Przykład.3 Wczytywanie całości pliku

```
<?php
$dane = fread(fopen("nazwa_pliku", "r"), filesize("nazwa_pliku"));
?>
```

Funkcja fread() będzie czytać dane bez przerwy – od początku pliku do końca, ignorując znaki końca linii – dla tej funkcji to po prostu zwykły znak.
Cały plik można wczytać też w jeszcze jeden sposób. Funkcja file($nazwa_pliku) zwraca tablicę, w której każdy element jest osobnym wierszem z pliku, którego nazwa jest podana w parametrze. Funkcję tą można wykorzystać do wczytania całego pliku do jednego stringa.

Przykład.4 Wczytywanie całości pliku przy użyciu funkcji file

```
<?php
$plik = implode('', file('nazwa_pliku'));
?>
```

Samo wczytanie pliku do tablicy może być przydatne. Traktując plik jako bazę danych, w której każdy wiersz to jeden rekord, wczytanie takiego pliku od razu do tablicy oszczędza wiele pracy przy ręcznym rozbijaniu pliku na wiersze.

3. Zapis do pliku

Dane do pliku można zapisać przy pomocy funkcji fputs(int wskaznik, string napis, int długość) lub funkcji frwite(int wskaźnik, zmienna). Jak przy większości funkcji operujących na plikach, niezbędne jest wskaźnika pliku na którym chcemy operować. Zapisać można albo całą zawartość zmiennej podanej jako drugi parametr, albo tylko do pewnej długości, którą to 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.

Przykład.5 Zapis do pliku

```
<?
// otwarcie pliku do odczytu
$plik = fopen("dane.txt", "r");

//odczytanie danych z pliku
$stareDane = fread($plik, filesize("dane.txt"));

// zamknięcie pliku
fclose($plik)

// stworzenie nowych danych
$noweDane  = "Nowe dane\n";
$noweDane .= $stareDane;

// zapisanie nowych danych

// otwarcie pliku do zapisu
$plik = fopen("dane.txt", "w");

// zapisanie danych
fputs($plik, $noweDane);
// lub fwrite($plik,$noweDane);
// zamknięcie pliku
fclose($plik);
?>
```

Zapis na koniec pliku jest łatwy – wystarczy otworzyć plik w trybie "a" i od razu można dodawać dane do pliku.

Zapis linia po linii do pliku:
fwrite($plik,$noweDane.PHP_EOL);

4. Przycinanie plików

PHP zawiera funkcję służącą do zmniejszania rozmiaru pliku do zadanej wielkości. Funkcja ftruncate(int wskaźnik, int rozmiar) służy właśnie do czegoś takiego. pamiętać, że wszelkie dane, które znajdują się powyżej podanego rozmiaru, będą utracone.

5. Blokowanie plików

Blokowanie plików jest jednym z ważniejszych zagadnień 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(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 blokade na pliku) i blokada wyłączna, zakładana jeśli plik ma być zapisywany. Drugi parametr funkcji może mieć takie wartości:

LOCK_SH - aby założyć blokadę dzieloną
LOCK_EX - aby założyć blokadę wyłączną
LOCK_UN - aby zdjąć każdą blokadę

Jak wykorzystać to w praktyce? 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.

Zależnie od zastosowania, można różnie reagować na niemożność założenia blokady. Można albo przerwać skrypt albo czekać na zdjęcie blokady. W przypadku licznika jedne odwiedziny w tą czy tamtą nie mają znaczenia w obliczu konieczności czekania na zakończenie działania innego skryptu – będą to ułamki sekund, ale bywa z tym różnie (zwłaszcza, jeśli takich czekających jest więcej). Natomiast jeśli chodzi o blokadę bazy danych, to czekanie na zdjęcie blokady jest koniecznością.

Przykład.6 Licznik odwiedzin z blokowaniem bazy

```
<?
// Nazwa pliku zawierającego licznik - względna do katalogu, w którym
// jest strona na której ma być licznik

$sciezka  = "licznik.dat";

// Próba otwarcia pliku do odczytu

if(!($plik=@fopen($sciezka, "r")))
{// Jeśli plik jeszcze nie istnieje, to jest inicjowany wartością 0
  $licz=0;
}
else
{ // jeśli istnieje, to jego wartość jest odczytywana a plik zamykany
  $licz = fgets($plik, 100);
  fclose($plik);
}

// Zwiększenie licznika o 1
$licz+=1;

// Otwarcie pliku do zapisu
$plik = fopen($sciezka, "w");

// Blokada - jeśli plik już jest zablokowany, to skrypt go zamyka i kończy działanie

if(!flock($plik, LOCK_EX))
{
  fclose($plik);
  return;
}
else
{ // Jeśli nie jest zablokowany, to następuje blokada i zapis danych
  fputs($plik, $licz);
  flock($plik, LOCK_UN);
  fclose($plik);
}

// Wyświetlenie stanu licznika
echo $licz;
?>
```

Ten plik możemy zapisać np. jako licz.php i w odpowiednim miejscu na stronie, gdzie ma się pojawić licznik, należy wpisać <?include "licz.php"?>. Jeśli pojawią się bledy w stylu „Permission denied” oznacza to, że coś jest nie tak z

prawami dostępu do pliku z danymi, a jeśli jeszcze on nie istnieje, to z prawami do katalogu gdzie ma być ten licznik. Serwer WWW powinien mieć prawa zapisu do tego pliku/katalogu.

6. 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. Chyba najlepiej je wypunktować.

Funkcje informacyjne

- fileatime($nazwa_pliku) - zwraca czas ostatniego odczytu pliku; czas ten jest zwracany w postaci timestamp (co to jest timestamp i jak tego używać – w następnym rozdziale)
- filectime($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. Kolejne indeksy zawierają:
    - urządzenie
    - i-węzeł
    - tryb zabezpieczenia i-węzła
    - liczba dowiązań
    - id właściciela
    - id grupy
    - typ urządzenia jeśli urządzenie jest oparte na i-węzłach
    - rozmiar w bajtach
    - czas ostatniego dostępu
    - czas ostatniej modyfikacji i-węzła
    - czas ostatniej zmiany
    - rozmiar bloku
    - liczba bloków

Jeśli funkcja ta jest wywołana na dowiązaniu symbolicznym, to informacje te będą dotyczyły pliku, na który wskazuje dowiązanie. Aby uzyskać informacje o samym dowiązaniu, należy użyć funkcji lstat() (zwracane wartości są takie same jak w przypadku funkcji stat())

Funkcje logiczne (zwracają wartość true lub false)

- 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

7. Operacje na plikach i katalogach

7a. Kopiowanie

Do kopiowania plików służe funkcja o jakże zaskakującej nazwie copy(). Pierwszy parametr to plik źródłowy a drugi to plik lub katalog docelowy. Funkcja ta zwraca wartość true jeśli kopiowanie się powiodło lub false w przeciwnym przypadku. Dobrze jest sprawdzać czy kopiowanie się powiodło.

Przykład.7 Sprawdzanie powodzenia kopiowania

```
<?php
copy($source, $destination) or die("Błąd przy kopiowaniu");
?>
```

7b. Przenoszenie i zmiana nazwy

Funkcja służaca do przenoszenia i zmiany nazwy pliku sugeruje tylko tą drugą funkcję, jednak bardzo dobrze spisuje się w obu zastosowaniach. Funkcja rename() pobiera 2 argumenty: nazwę pliku źródłowego oraz nazwę pliku docelowego jeśli funkcja ma zmienić nazwę, nazwę katalogu jeśli plik ma być przeniesiony, lub ścieżkę wraz z nową nazwą,
jeśli plik ma być przeniesiony ze zmianą nazwy.

Przykład.8 Sposoby użycia funkcji rename

```
<?
rename("aaa", "bbb");     // zmiana nazwy pliku "aaa" na "bbb"
rename("bbb", "test/");   // przeniesienie pliku "bbb" do katalogu "test"
rename("aaa", "..");      // przeniesienie pliku "aaa" do katalogu nadrzędnego
rename("aaa", "test/bbb"); // przeniesienie pliku "aaa" do katalogu "test" ze zmianą nazwy na "bbb"
?>
```

7c. Usuwanie

Usuwanie plików jest rzeczą sprawiającą najwięcej trudności początkującym programistom PHP. Dzieje się tak głównie dlatego, że funkcja usuwająca pliki nazywa się unlink ($nazwa_pliku), co jest wynikiem tego, że PHP powstało początkowo pod systemy Uniksowe. Funkcja ta może nie działać pod systemami Windows – tam najoczywistszym rozwiązaniemjest wywołanie programu del z odpowiednim parametrem.
Do usuwania katalogów służy funkcja rmdir($nazwa).

7d. Tworzenie katalogów

Tworzeniem katalogów zajmuje się funkcja mkdir($nazwa_katalogu, $tryb). Drugi parametr określa tryb utworzenia katalogu (prawa dostępu). Funkcja zwraca wartość true jeśli katalog został utworzony lub false w przeciwnym przypadku.

8. Przetwarzanie ścieżki

PHP posiada wiele funkcji do analizy i przetwarzania zmiennych tekstowych, które zawierają ścieżkę do pliku. Funkcja basename($path) zwraca nazwę pliku a dirname($path) wszystko oprócz nazwy pliku. Z kolei funkcja pathinfo($path) zwraca tablicę asocjacyjną zawierającą informację o ścieżce podanej jako parametr – klucz „dirname” zawiera nazwę katalogu, „basename” nazwę pliku a „extension” – rozszerzenie pliku.

Przykład.9 Przetwarzanie ścieżki

```
<?php
$path = "c:/users/student/httpd/html/index.html";
echo dirname($path);    // wyświetli "/home/httpd/html"
echo basename($path);   // wyświetli "index.html"
$arr = pathinfo($path);
echo $arr["dirname"];   // wyświetli "/home/httpd/html"
echo $arr["basename"];  // wyświetli "index.html"
echo $arr["extension"]; // wyświetli "html"
```

```
?>
```

9. Operacje na katalogach

Istnieją 2 mechanizmy przeglądania zawartości katalogów. Pierwszy to mechanizm oparty na dwóch funkcjach: opendir($katalog) i readdir($handle). Funkcja opendir($katalog) otwiera katalog o podanej nazwie i zwraca uchwyt do niego. Z kolei funkcja readdir($handle) za każdym wywołaniem zwraca nazwę kolejnego pliku/katalogu z otwartego katalogu.

Przykład.10 Wyświetlenie wszystkich nazw plików z katalogu /tmp

```
<?php
if ($dir = @opendir("/tmp"))
{
  while($file = readdir($dir))
 {
    echo "$file\n";
  }
  closedir($dir);
}
?>
```

Drugi sposób oparty jest na mechanizmie pseudo-obiektowym.

Przykład.11 Odczytywanie zawartości katalogu przy uzyciu funcji dir

```
<?php
$d = dir("/katalog");
echo "Handle: ".$d->handle."<br>\n";
echo "Path: ".$d->path."<br>\n";
while($entry=$d->read()) {
  echo $entry."<br>\n";
}
$d->close();
?>
```

Obiekt zwraceny przez funkcję dir($katalog) posiada 3 metody: metoda handle() zwraca zwykły uchwyt, który może być używany w funkcji readdir(), metoda path() zwraca string zawierający ścieżkę do katalogu a metoda read() zwraca kolejne nazwy plików/katalogów, podobnie jak funkcja readdir().

Dla funkcji readdir() i metody read() nie ma różnicy między plikiem lub katalogiem – zwracane są wszystkie nazwy z katalogu. A czy to jest katalog programista musi sam sprawdzić za pomocą funkcji is_dir().

ZADANIA

1. Skrypt wczytuje dwie liczby z formularza i zapisuje do pliku dane1.txt:
   a) liczby w osobnych liniach
   b) liczby w tej samej linii
   c) sumę i iloczyn liczb
2. Skrypt zapisuje do pliku dane2.txt 10 liczb (losowe lub z klawiatury).
3. Skrypt wczytuje liczby z pliku dane2.txt a następnie szuka największej z nich oraz liczy średnia i wpisuje do pliku dane3.txt : max = ….. ; srednia = ……
4. Skrypt zapisuje do pliku dane4.txt 5 stolic Europy w tej samej linii oraz w osobnych liniach.
5. Skrypt wczytuje dane z pliku dane4.txt i liczy ile znaków było w każdym wierszu.