4. Pliki

4.1. Informacje ogólne o dostępie do plików w PHP

Jak praktycznie każdy język programowania, PHP zapewnia dostęp do systemu plików znajdujących się na komputerze, na którym uruchamiany jest skrypt, bądź też zdalnych.Interfejs dostępu do plików w PHP jest zbliżony do tego, co udostępnia biblioteka standardowa języka C, znajdziemy jednak kilka specyficznych dla PHP, a dość wygodnych funkcji.

Plik w PHP jest reprezentowany przez zmienną typu **zasób** (resource). Większość funkcji związanych z plikami wymaga takiej zmiennej jako pierwszego parametru. Istnieją jednak funkcje, które używają jedynie ścieżki dostępu do pliku.

4.2. Sprawdzanie istnienia pliku – *file_exists()*

Zanim spróbujemy plik otworzyć, warto sprawdzić jego istnienie. Funkcja *file_exists()* przyjmuje jako argument łańcuch tekstowy, będący ścieżką dostępu do pliku, wynikiem jest TRUE lub FALSE. Oto przykład:

```
if (file_exists('dane/wyniki.txt'))
{
   // plik istnieje
}
```

Ścieżka dostępu do pliku, jeżeli jest względna, jest interpretowana w odniesieniu do katalogu, w którym znajduje się skrypt. Jeżeli mamy następującą stukturę katalogów:

```
/
    /home
        /user6
            /public_html
                /skrypty
                    skrypt.php
                /dane
                    dane3.txt
```

to z wnętrza skryptu „skrypt.php” możemy się odwołać do pliku „dane3.txt” albo bezwzględnie, poprzez „/home/user6/public_html/dane/dane3.txt” albo względnie poprzez „../dane/dane3.txt”.

4.3. Wczytanie pliku do łańcucha tekstowego – *file_get_contents()*

Funkcja ta wczytuje plik o podanej nazwie (całą zawartość) do jednego łańcucha tekstowego, który może być rzecz jasna dość długi.

```
$text = file_get_contents('../dane/dane3.txt');
```

Warto zwrócić uwagę, że o ile w pliku występują znaki końca linii, to zostaną one bez zmian umieszczone w łańcuchu. Można tą funkcją ładować pliki binarne, ale trzeba się liczyć z tym, że łańcuch może zawierać dowolne znaki z całego zakresu 0 – 255.

4.4. Wczytanie pliku do tablicy – *file()*

Jest to funkcja przeznaczona do ładowania plików tekstowych podzielonych na linie (zakończone znakiem końca linii). Każda linia tekstu zostaje załadowana do kolejnego elementu tablicy. Klucze kolejnych elementów tworzone są automatycznie (liczby całkowite 0, 1, 2...). Wartości kolejnych elementów to łańcuchy tekstowe zawierające zawartość kolejnych linii tekstu **wraz ze znakami końca linii** (nie są one odrzucane). Można się ich pozbyć wykonując na elementach funkcję *rtrim()*.

```
$tablica = file('../dane/dane3.txt');
```

4.5. Badanie rozmiaru pliku - *filesize()*

Funkcja podaje rozmiar pliku o podanej nazwie w bajtach:

```
$dlugosc = filesize('../dane/dane3.txt');
```

4.6 Badanie daty ostatniej modyfikacji pliku – *filemtime()*

Funkcja zwraca datę ostatniej modyfikacji pliku o podanej nazwie. Data ta jest zwracana w postaci ilości sekund jaka upłynęła od 1 stycznia 1970, 00:00:00 do chwili modyfikacji pliku. Można ją przetworzyć w postać czytelną dla człowieka za pomocą funkcji *date()*.

4.7. Otwieranie pliku – *fopen()*

Jest to pierwsza z serii funkcji posługujących się tzw. uchwytem pliku – zmienną typu **zasób**. *fopen()* tworzy taką zmienną w czasie otwierania pliku. Funkcje używające uchwytu pliku są z reguły zgodne ze swoimi odpowiednikami ze standardowej biblioteki języka C. Plik może zostać otwarty w jednym z 8 trybów, oto one wraz z krótkim opisem:

r	tryb tylko do odczytu
r+	zapis/odczyt od początku istniejącego pliku
w	tylko zapis, nowy plik, ewentualna stara zawartość pliku usunięta
w+	zapis/odczyt, nowy plik, ewentualna stara zawartość pliku usunięta
a	zapis na końcu istniejącego pliku lub utworzenie nowego
a+	zapis/odczyt na końcu istniejącego pliku lub utworzenie nowego
x	zapis tylko i wyłącznie do nowego pliku (FALSE jeżeli już istnieje)
x+	zapis/odczyt tylko i wyłącznie do nowego pliku (FALSE jeżeli już istnieje)

```
if ($plik = fopen('../dane/dane3.txt', 'r'))
{
   // robimy coś z plikiem
   fclose($plik);
}
```

4.7. Zamykanie pliku – *fclose()*

Funkcja ta zamyka plik o podanym uchwycie (a więc plik otwarty uprzednio za pomocą *fopen()*). Czy należy zawsze zamykać pliki? W zasadzie niekoniecznie, ponieważ PHP samo zamyka wszystkie niezamknięte pliki w momencie zakończenia skryptu. Zamykanie plików zaraz po zakończeniu ich używania jest jednak dobrą praktyką, ponieważ zmniejsza zużycie zasobów serwera (przypominam o limicie przydziału pamięci dla skryptu). Poza tym zawartość niezamkniętego ręcznie pliku może być nieokreślona w przypadku anormalnego zakończenia skryptu.

4.8. Czytanie fragmentu pliku – *fread()*

Funkcja odczytuje fragment pliku od bieżącej pozycji i umieszcza go w łańcuchu tekstowym. Długość fragmentu jest podana w bajtach, znaki końca linii traktowane są jak wszystkie inne.

```
$fragment = fread($plik, 47); // $fragment zawiera 47 bajtów z pliku
```

W przypadku dojścia do końca pliku zostanie oczywiście mniej bajtów.'

4.9. Czytanie pojedynczego znaku – *fgetc()*

Funkcja jako swój wynik zwraca jeden znak odczytany z pliku.

```
$znak = fgetc($plik);
```

W przypadku dojścia do końca pliku funkcja zwróci wartość logiczną FALSE.

4.10. Czytanie linii tekstu – *fgets()*

Jak łatwo się domyślić, ta funkcja służy do czytania z plików tekstowych. Wynikiem działania funkcji jest łańcuch tekstowy zawierający wczytaną z pliku linię. Linia jest czytana do momentu napotkania znaku końca linii, albo (opcjonalnie) po przeczytaniu maksymalnej podanej liczby znaków. Należy zwrócić uwagę na istotny fakt – jeżeli czytanie zakończy się po napotkaniu znaku końca linii, to znak ten jest dołączony na końcu łańcucha, jeżeli natomiast w grę wejdzie iograniczenie ilości znaków, to znaku końca linii nie będzie. Podobnie może się wydarzyć, jeżeli ostatnia linia w pliku nie ma znaku końca linii (wtedy czytanie zatrzyma się na EOF).

```
$linia = fgets($plik);      // czytaj całą linię
$linia = fgets($plik, 80);  // czytaj linię, ale nie więcej niż 80 znaków
```

4.11. Czytanie linii z odrzuceniem tagów HTML i PHP – *fgetss()*

Ta z kolei funkcja działa bardzo podobnie do fgets() ale usuwa z ładowanego tekstu wszystkie tagi języka HTML i kod PHP. Opcjonalnie możemy podać, jakie tagi życzymy sobie zostawić. Najlepiej zilustrują to przykłady:

```
// czytanie linii tekstu z usunięciem kodu PHP i wszystkich znaczników HTML

$linia = fgetss($plik);  // tylko w PHP5, w PHP4 trzeba podać ograniczenie

// to samo, ale z ograniczeniem długości linii

$linia = fgetss($plik, 80);

//  z ograniczeniem długości, ale pozostaw proste formatowanie

$linia = fgetss($plik, 80, '<b>,</b>,<i>,</i>,<u>,</u>,<br>');
```

Funkcja *fgetss()* może być szczególnie użyteczna, gdy jest używana do przetwarzania kodu HTML np. pobranego z innej strony internetowej. Jest to możliwe, gdyż *fopen()* pozwala na otworzenie jako pliku dowolnego zasobu dostępnego poprzez protokół HTTP. Więcej na ten temat w punkcie 4.14.

4.12. Zapis do pliku – *fwrite()*

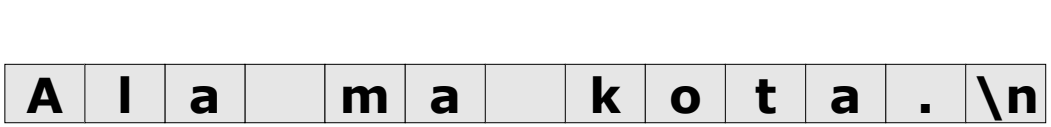
W przypadku zapisu nie ma już tylu funkcji co przy odczycie, jest po prostu jedna funkcja zapisująca wskazany łańcuch tekstowy jako ciąg bajtów do pliku. Jest oczywiste, że w ten sposób można zapisać również linię tekstu, o ile na końcu łańcucha będzie znak końca linii. Funkcja pozwala też na opcjonalne ograniczenie długości zapisywanego łańcucha do określonej liczby bajtów.

```
// zapis bez ograniczenia

$linia = "To jest pierwsza linia\n";
fwrite($plik, $linia);
fwrite($plik, "To jest druga linia\n");

// z ograniczeniem

fwrite($plik, "A to trzecia\n", 6);
// uwaga, znak końca linii zostanie obcięty!
```

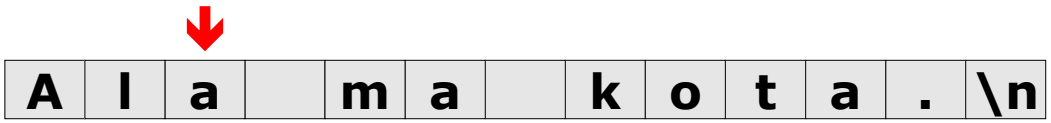
4.13. Poruszanie się po pliku funkcją *fseek()*

Funkcja *fseek()* służy do zmiany aktualnego miejsca czytania i zapisu w pliku. Przy otwarciu pliku miejsce to automatycznie znajduje się na jego początku (w niektórych trybach otwarcia pliku na końcu, patrz punkt 4.7.). Funkcje czytające i zapisujące dane przesuwają to miejsce na koniec odczytanych lub zapisanych danych. Funkcja *fseek()* pozwala na ręczne manipulacje. Miejsce czytania i zapisywania (zwane też wskaźnikiem pliku) może zostać przesunięte w nowe położenie określone względem początku (SEEK_SET), końca (SEEK_END) lub aktualnego położenia (SEEK_CUR) w pliku. Należy pamiętać o tym, że *fseek()* może zachować się w różny sposób przy pracy z sieciowymi systemami plików, albo z plikami otwartymi poprzez protokół HTTP.

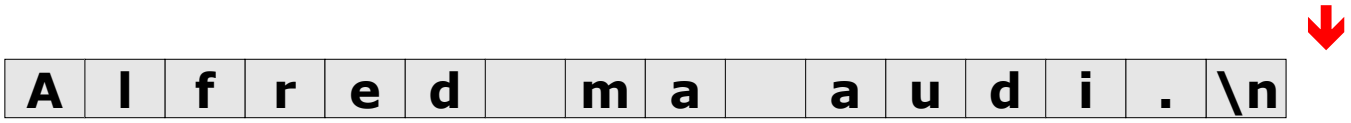
```
$plik = fopen('plik.txt','x+');
fwrite($plik, "Ala ma kota.\n”);
```



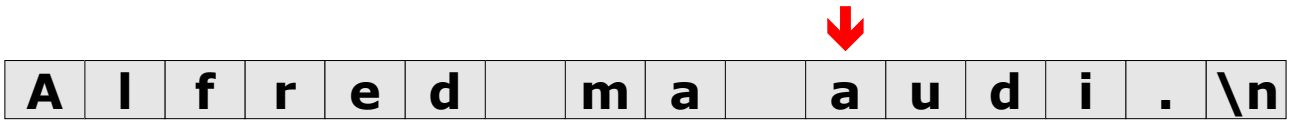

```
fseek($plik, 2, SEEK_SET);
```



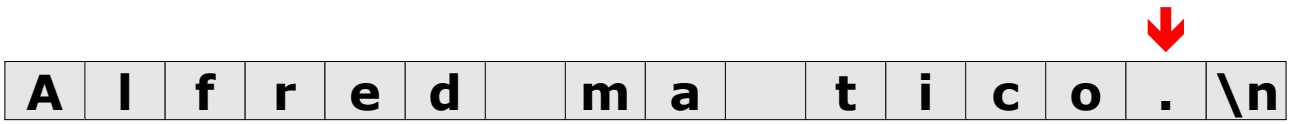

```
fwrite($plik, "fred ma audi.\n");
```




```
fseek($plik, -6, SEEK_END);
```




```
fwrite($plik, "tico turbo glx", 4);
```



4.14. Strumienie HTTP jako pliki

PHP umożliwia traktowanie zasobów dostępnych przez protokół HTTP jak zwykłych plików.Należy jednak zwrócić uwagę na szereg ograniczeń. Pliki otwarte w ten sposób są tylko do odczytu, nie działa *fseek()* i większość funkcji podających atrybuty pliku. Głównym przeznaczeniem takich plików jest po prostu sekwencyjne odczytanie zawartości. Używanie strumieni HTTP jako plików jest możliwe jeżeli ustawiona jest zmienna konfiguracyjna *allow_url_fopen*. PHP obsługuje również strumienie zaszyfrowane (protokół HTTPS).

Przykład wyświetla kod źródłowy strony otwartej jako plik.

```
$plik = fopen('http://jakis.serwer.pl/stronka.html','r');
$tab1 = array('<','>","\n");
$tab2 = array('&lt;','&gt;','<br>');

if ($plik)
{
   while($linia = fgets($plik))
   {
      $l2 = str_replace($tab1, $tab2, $linia);
      echo $l2;
   }
}
```