

Лекция 10

Работа с файлами в РНР

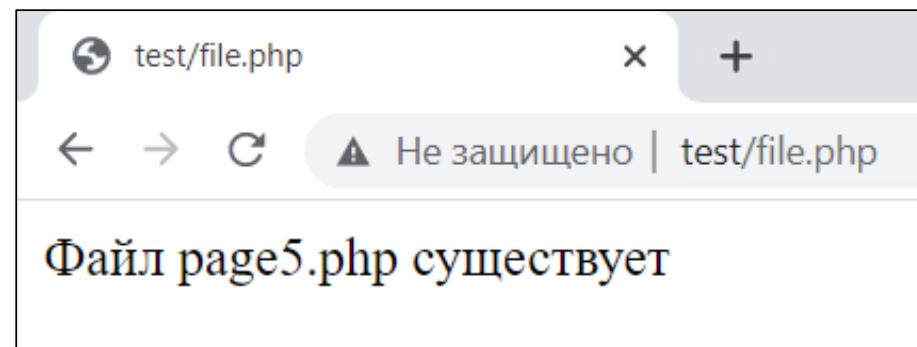
Анализ файлов

PHP содержит множество функций, дающих информацию о файлах.

file_exists() - проверяет существование указанного файла или каталога

file_exists(string \$filename): bool

```
<?php
$filename = 'page5.php';
if (file_exists($filename)) {
    echo "Файл $filename существует";
} else {
    echo "Файл $filename не существует";
}
?>
```

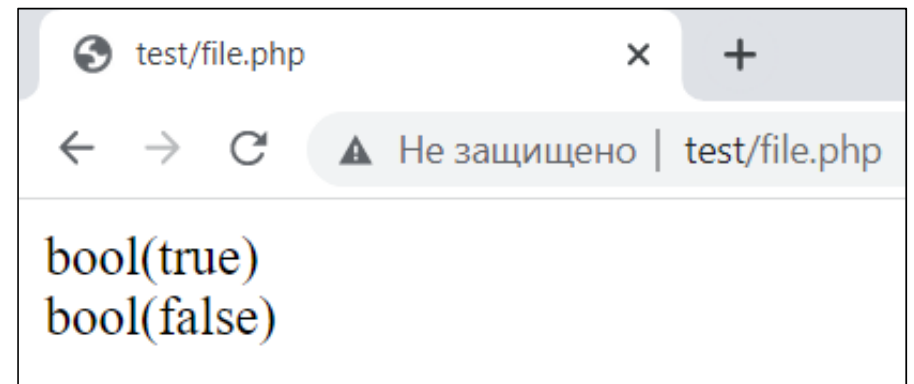


Анализ файлов

is_file() - определяет, является ли файл обычным файлом.

is_file(string \$filename): bool

```
<?php
var_dump(is_file('hello.txt'));
var_dump(is_file('/domains/test/'));
?>
```

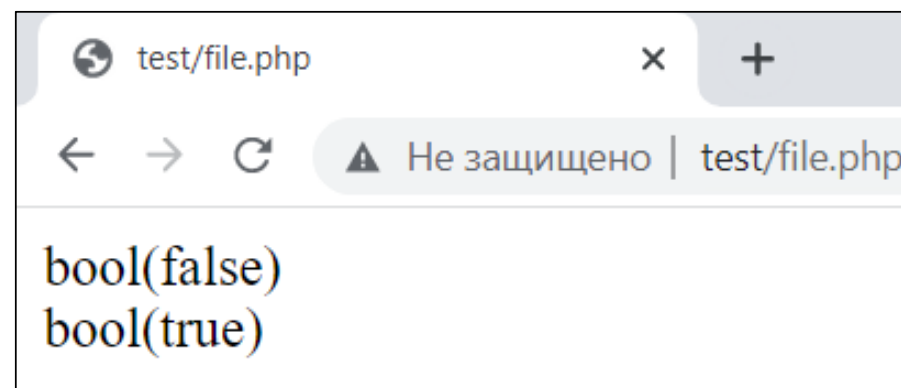


Анализ файлов

is_dir() - определяет, является ли имя файла директорией.

is_dir(string \$filename): bool

```
<?php
var_dump(is_dir('hello.txt'));
var_dump(is_dir('/domains/test/'));
?>
```

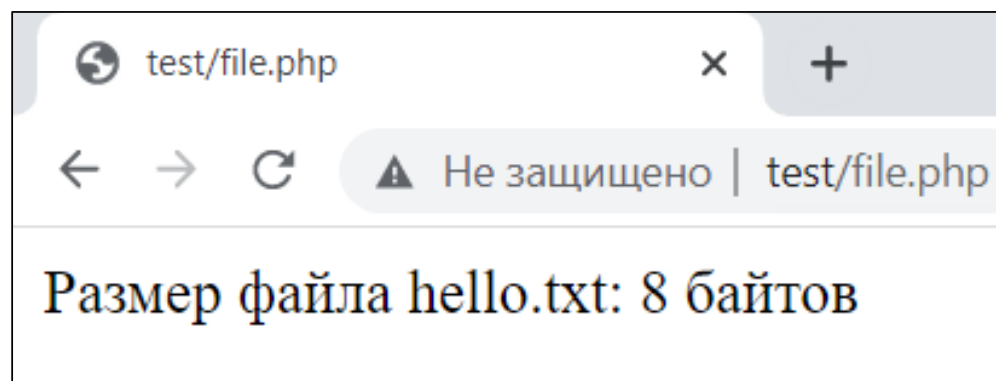


Анализ файлов

filesize() — возвращает размер файла.

filesize(string \$filename): int|false

```
<?php
$filename = 'hello.txt';
echo 'Размер файла ' . $filename . ': ' . filesize($filename) . ' байтов';
?>
```



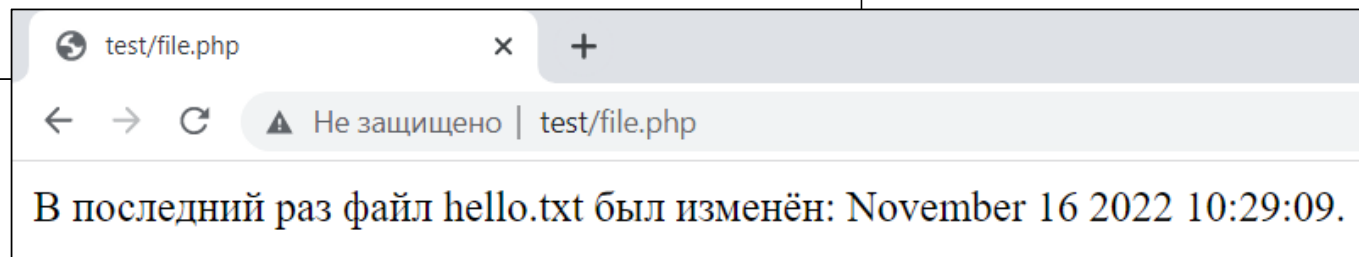
Анализ файлов

filemtime() – возвращает время последнего изменения файла.

filemtime(string \$filename): int|false

Возвращает время последнего изменения указанного файла или false в случае возникновения ошибки. Время возвращается в формате временной метки Unix, который подходит для передачи в качестве аргумента функции date().

```
<?php
$filename = 'hello.txt';
if (file_exists($filename)) {
    echo "В последний раз файл $filename был изменён: " .
        date ("F d Y H:i:s.", filemtime($filename));
}
?>
```



Анализ файлов

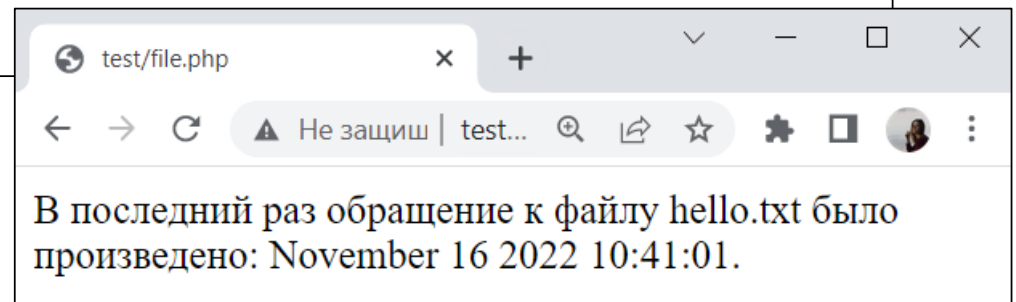
filetime() – возвращает время последнего доступа к файлу.

filetime(string \$filename): int|false

Возвращает время, когда в последний раз был осуществлён доступ к указанному файлу или false в случае возникновения ошибки. Время возвращается в виде временной метки Unix.

```
<?php
$filename = 'hello.txt';
if (file_exists($filename)) {
    echo "В последний раз обращение к файлу $filename было
произведено: " . date("F d Y H:i:s.", filetime($filename));
}
?>
```

Внимание! С удаленными файлами эти функции не работают. Их можно применять только к локальной файловой системе.



Анализ файлов

feof() – проверяет, достигнут ли конец файла.

feof(resource \$stream) : bool

Возвращает true, если указатель файла указывает на EOF или произошла ошибка (в том числе превышено время ожидания сокета), иначе возвращает false.

Управление файлами

touch() – устанавливает время доступа и модификации файла.

```
touch(string $filename, $mtime = null, $atime = null): bool
```

Пытается установить время доступа и модификации файла с именем **filename** в значение **mtime**. Обратите внимание, что время доступа изменяется всегда, независимо от количества аргументов.

Если файл не существует, он будет создан.

Управление файлами

`touch(string $filename, $mtime = null, $atime = null): bool`

```
<?php
```

```
// Это время изменения, установим его на час назад.
```

```
$time = time() - 3600;
```

```
if (!touch('hello.txt', $time)) {
```

```
    echo 'Упс, что-то пошло не так...';
```

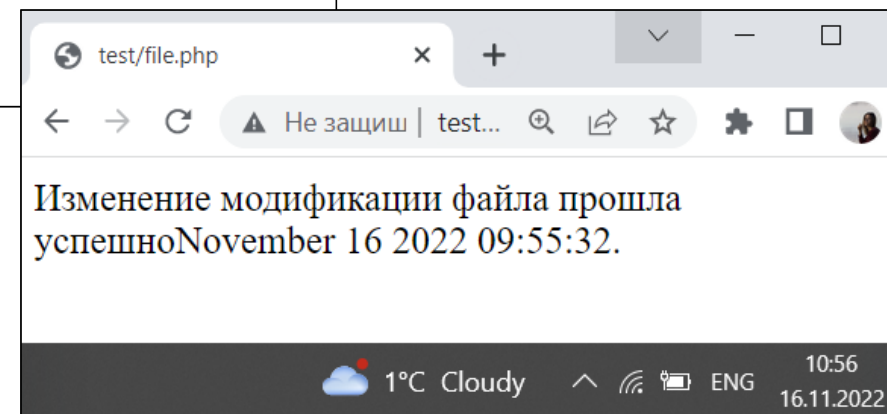
```
} else {
```

```
    echo 'Изменение модификации файла прошла успешно'.
```

```
date ("F d Y H:i:s.", filemtime('hello.txt'));
```

```
}
```

```
?>
```



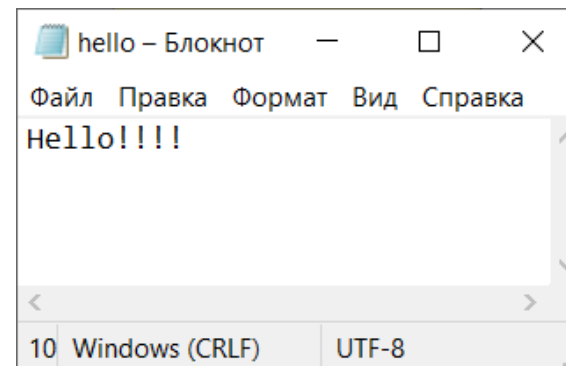
Управление файлами

copy() – копирует файл.

copy(string \$from, string \$to): bool

Копирует файл **from** в файл с именем **to**. Если **to** является URL, то операция копирования может завершиться ошибкой, если обёртка URL не поддерживает перезаписывание существующих файлов.

Внимание! Если целевой файл уже существует, то он будет перезаписан.

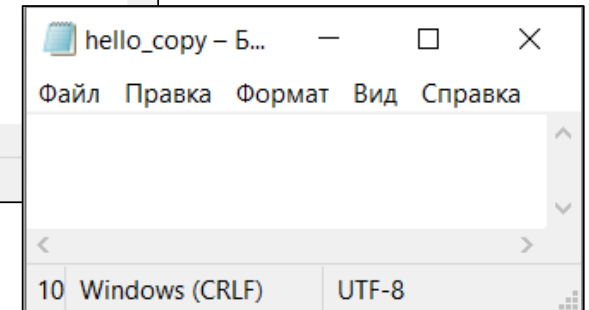
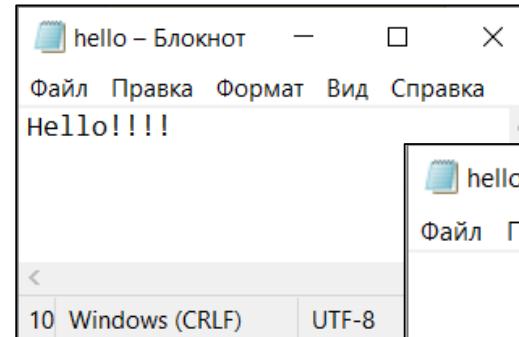


Управление файлами

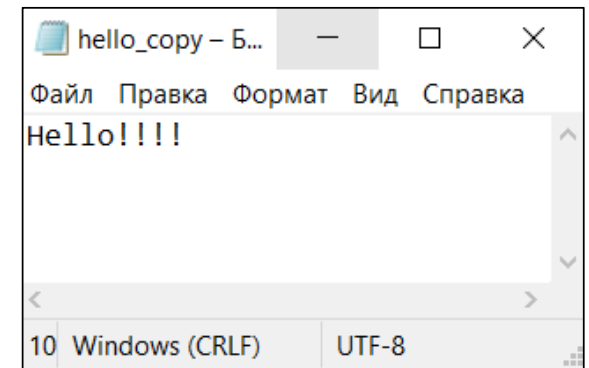
```
copy(string $from, string $to): bool
```

```
<?php
$file = 'hello.txt';
$newfile = 'hello_copy.txt';

if (!copy($file, $newfile)) {
    echo "не удалось скопировать $file...\n";
}
?>
```



После копирования



Управление файлами

rename() – переименовывает файл или директорию.

rename(string \$from, string \$to): bool

Пытается переименовать **from** в **to**, перенося файл между директориями, если необходимо. Если **to** существует, то он будет перезаписан. При переименовании директории с существующим **to** будет выведено предупреждение.

```
<?php
rename("hello.txt", "1hello2.txt");
?>
```

Управление файлами

unlink() – удаляет заданный файл.

unlink(string \$filename): bool

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

Открытие файла

Обычно работа с файлами разделяется на 3 этапа:

1. Открытие файла.
2. Операции с данными.
3. Закрытие файла.

Для того чтобы открыть файл в среде PHP, используется функция `fopen()`. Обязательными параметрами этой функции является **имя файла** и **режим файла**.

`fopen(string $filename, string $mode)`

`$filename` – имя файла или абсолютный путь к нему, если файл находится не в текущем каталоге

`$mode` – режим открытия файла

Открытие файла

fopen(string \$filename, string \$mode)

Режимы :

r - файл открывается только для чтения. Если файла не существует, вызов регистрирует ошибку. После удачного открытия указатель файла устанавливается на его первый байт, то есть на начало.

r+ - файл открывается одновременно на чтение и запись. Указатель текущей позиции устанавливается на первый байт. Если файла не существует, возвращается false. Если в момент записи указатель файла установлен где-то по середине файла, то данные запишутся прямо поверх уже имеющихся, при необходимости увеличив размер файла.

w - создает новый пустой файл. Если на момент вызова файл с таким именем уже существовал, то он предварительно уничтожается. В случае неверно заданного имени файла, вызов завершается неудачей.

Открытие файла

`fopen(string $filename, string $mode)`

w+ - аналогичен ***r+***, но если файл изначально не существовал, он создается. После этого с файлом можно работать как в режиме чтения, так и в режиме записи. Если файл существовал до момента вызова, его содержимое удаляется.

a - открывает существующий файл в режиме записи, и при этом сдвигает указатель текущей позиции за последний байт файла. Вызов неуспешен в случае отсутствия файла.

a+ - открывает файл в режиме чтения и записи, указатель файла устанавливается на конец файла, при этом содержимое файла не уничтожается. Отличается от “***a***” тем, что если файла изначально не существовало, то он создается. Этот режим полезен, если нужно что-то дописать в файл, но неизвестно, создан ли уже такой файл.

b - флаг, указывающий на работу (чтение и запись) с двоичным файлом.

Открытие файла

`fopen(string $filename, string $mode)`

```
<?php
```

```
// Открывает файл на чтение
```

```
$fp = fopen("hello.txt", "r");
```

```
// Открывает рисунок таким же образом, только с флагом b
```

```
$fp = fopen("clouds.png", "rb");
```

```
// Открывает HTTP соединение на чтение
```

```
$fp = fopen("http://www.yandex.ru", "r");
```

```
//Открываем FTP соединение с указанием логина и пароля
```

```
$fp = fopen("ftp://user:password@example.ru", 'w');
```

```
?>
```

Операции с данными файла

Записывать данные в файл при помощи PHP можно при помощи функции **fwrite()**.

Это функция принимает 2 обязательных параметра и 1 необязательный. В качестве обязательных параметров выступает дескриптор файла и режим файла:

```
fwrite(resource $fp, string $data, [int $length = null]): int|false
```

Список параметров:

fp - указатель (resource) на файл, обычно создаваемый с помощью функции `fopen()`.

data - записываемая строка.

length - если параметр `length` является целым числом (int), запись остановится после того, как `length` байтов будут записаны или будет достигнут конец строки `data`, смотря что произойдёт раньше.

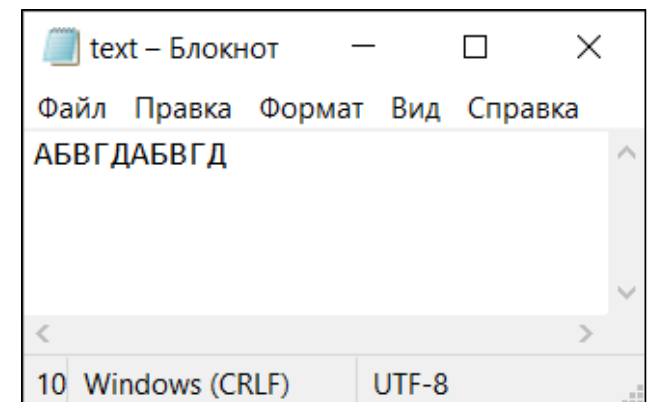
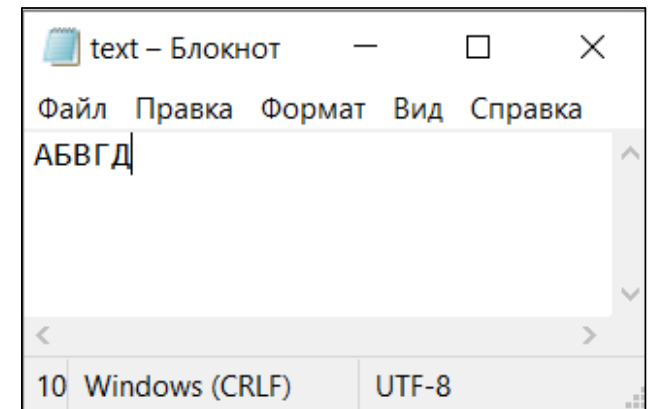
Операции с данными файла

`fwrite(resource $fp, string $data, [int $length = null]): int|false`

```
<?php
$fp = fopen("text.txt", "w");
fwrite($fp, "АБВГД");
?>
```

В результате работы функции в начало файла "text.txt" была записана строка "АБВГД". Если файл не существовал, то будет создан новый файл "text.txt".

Повторение данного вызова приведет к тому, что в конец файла будут записаны те же пять байт, и в файл будет записана строка "АБВГДАБВГД". Это также связано с работой указателя позиции в файле.

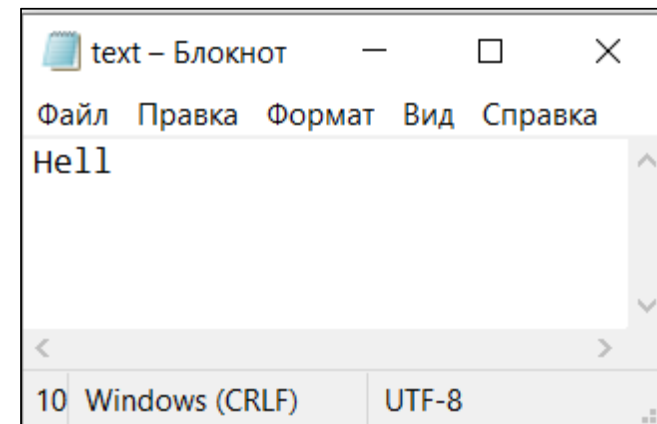


Операции с данными файла

`fwrite(resource $fp, string $data, [int $length = null]): int|false`

Если в данном примере указать в качестве третьего аргумента целое значение **`$length`**, то функция прекратит работу сразу после записи **`$length`** байт. Если строка содержит менее чем **`$length`** байт, то она будет записана в файл полностью.

```
<?php
$fp = fopen("text.txt", "w");
fwrite($fp, "Hello World!", 4);
?>
```



Операции с данными файла

```
fread(resource $fp, int $length): int|false
```

\$fp – указатель (resource) на файл, обычно создаваемый с помощью функции `fopen()`.

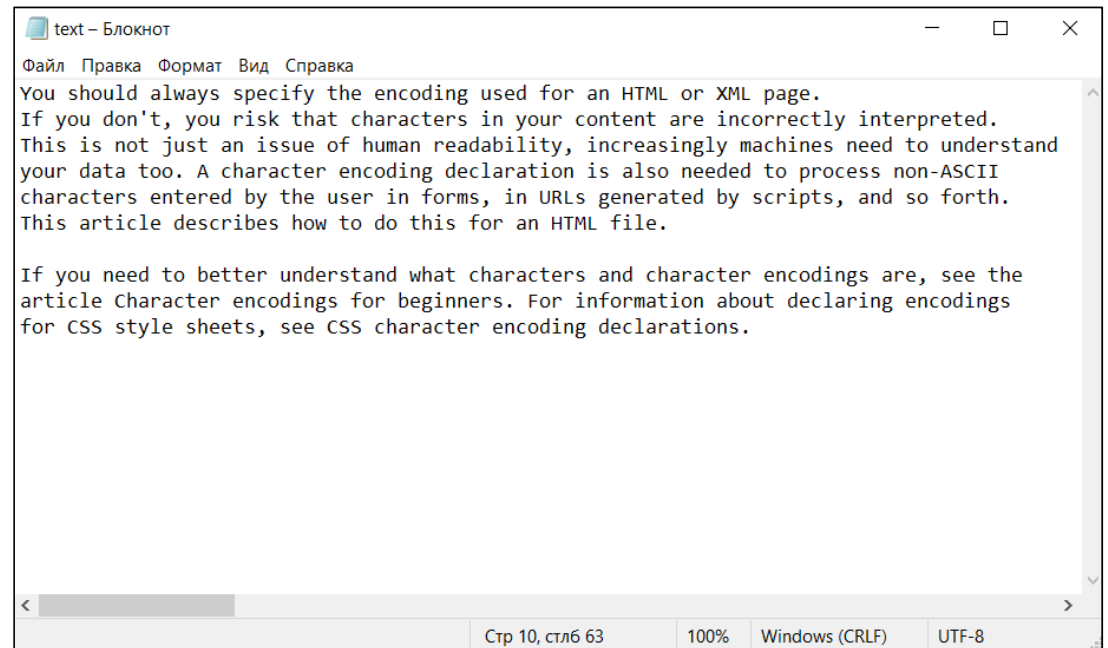
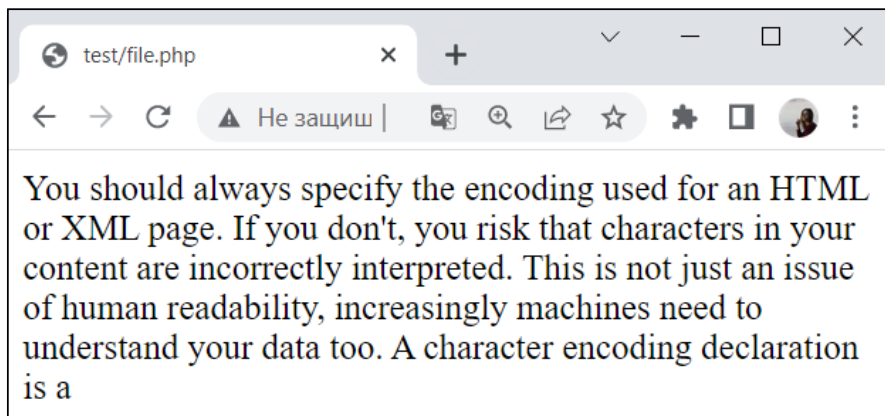
\$length – количество символов строки.

Функция **fread** читает из файла, заданного дескриптором **\$fp**, число **\$length** символов и возвращает строку этих символов. После чтения указатель файла продвигается к следующей после прочитанного блока позиции. Если число **\$length** больше содержимого файла, возвращается та информация, которую удалось считать.

Операции с данными файла

`fread(resource $fp, int $length): int|false`

```
<?php
$fp = fopen("text.txt", "r");
$data = fread($fp, 300);
echo $data;
?>
```



Операции с данными файла

`fgets(resource $fp, int $length): int|false`

Читает из файла строку, заканчивающуюся символом новой строки.

`$fp` – указатель (**`resource`**) на файл, обычно создаваемый с помощью функции **`fopen()`**,
`$length` – количество символов.

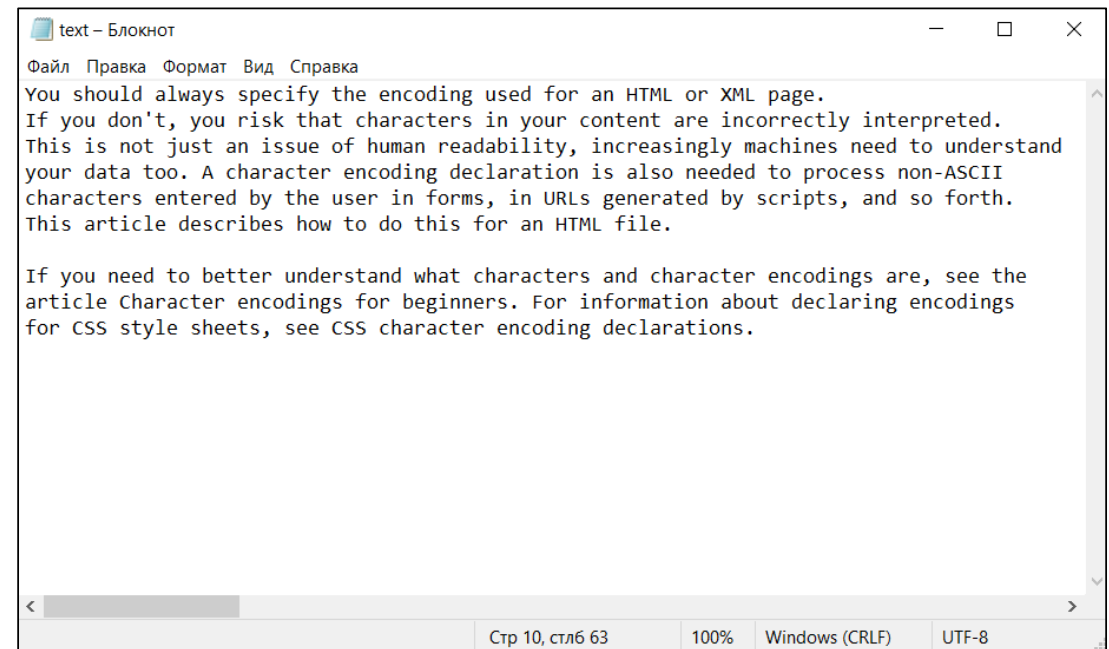
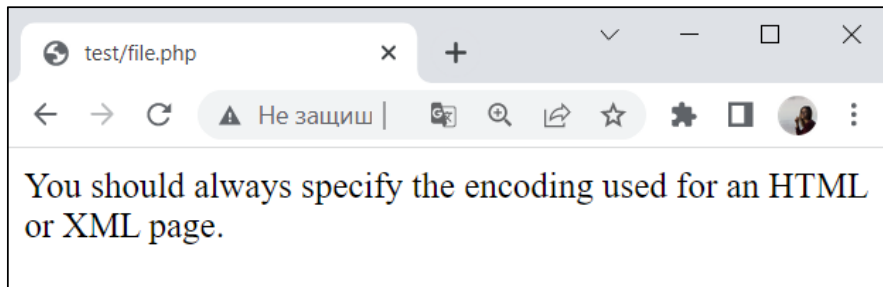
Функция **`fgets`** читает и возвращает строку длиной **`$length-1`** байт из файла **`$fp`**, переданного в ее параметрах. Если строка в файле занимает больше **`$length-1`** байт, то возвращаются только ее **`$length-1`** символов.

При достижении конца файла, функция возвращает пустую строку. Символ новой строки также считывается и включается в результат.

Операции с данными файла

fgets(resource \$fp, int \$length): int|false

```
<?php
$fp = fopen("text.txt", "r");
$data = fgets($fp, 300);
echo $data;
?>
```



Различие между **fgets** и **fread** заключается в том, что **fgets** прекращает чтение после того, как был достигнут конец строки или считано **\$length - 1** байт, тогда как функция **fread** игнорирует конец строки и считывает **\$length** байт.

Операции с данными файла

`fgetc(resource $fp) : string|false`

Считывает символ из переданного указателя на файл.

Функция **`fgetc`** принимает один аргумент, указатель (`resource`) на файл, `$fp`, и возвращает только один символ из связанного с данным дескриптором файла. Если функция достигает конца файла, то она возвращает `false`.

При этом код

```
$one_char = fgetc($fp);
```

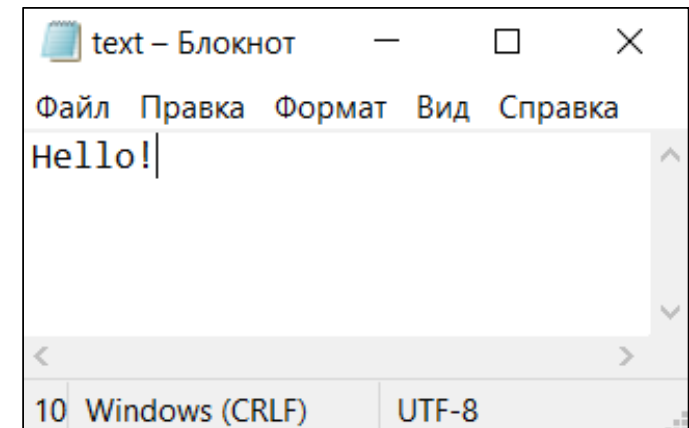
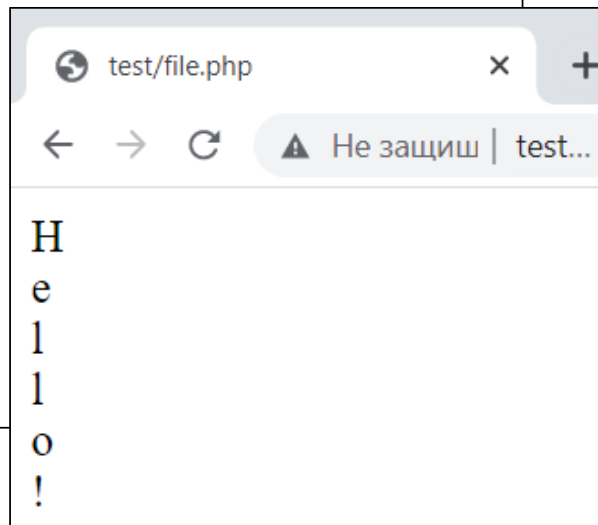
эквивалентен данному коду:

```
$one_char = fread($fp, 1);
```

Операции с данными файла

`fgetc(resource $fp): string|false`

```
<?php
$file = "text.txt";
if(!($fp = fopen($file, "r")))
{
    echo "Невозможно открыть файл $file.";
}
$result = "";
while (!feof($fp))
{
    $ch = fgetc($fp);
    echo "$ch<br>";
}
?>
```



Операции с данными файла

```
readfile(string $filename)
```

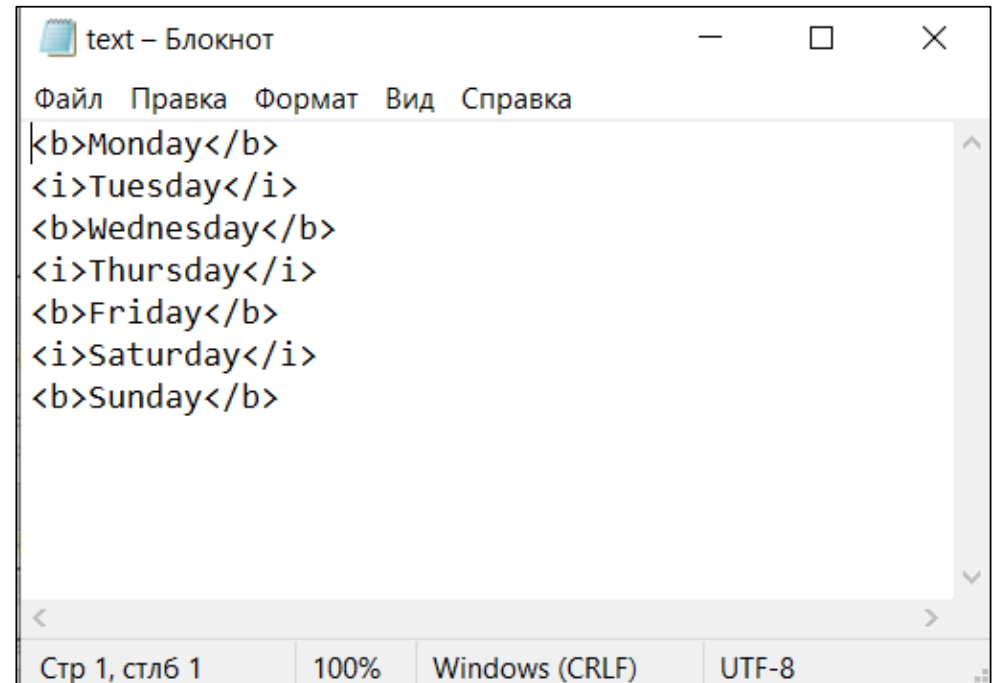
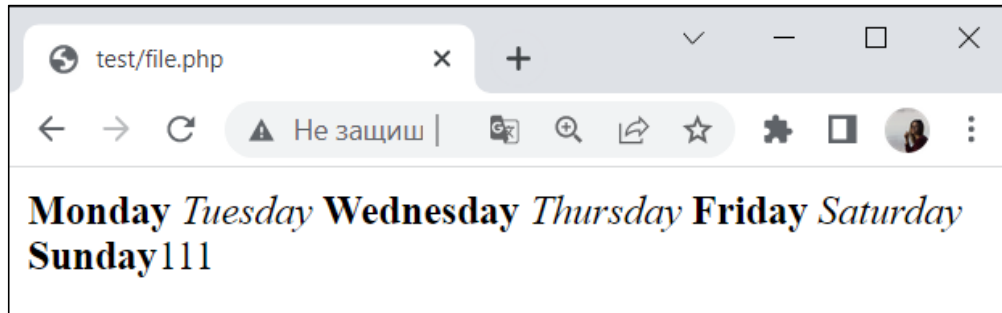
Считывает файл как единое целое, принимает один обязательный параметр (имя файла). Функция открывает файл, отображает его содержимое в окне браузера, а затем закрывает файл.

Возвращает количество прочитанных из файла байт в случае успешного выполнения или `false` в случае возникновения ошибки.

Операции с данными файла

`readfile(string $filename)`

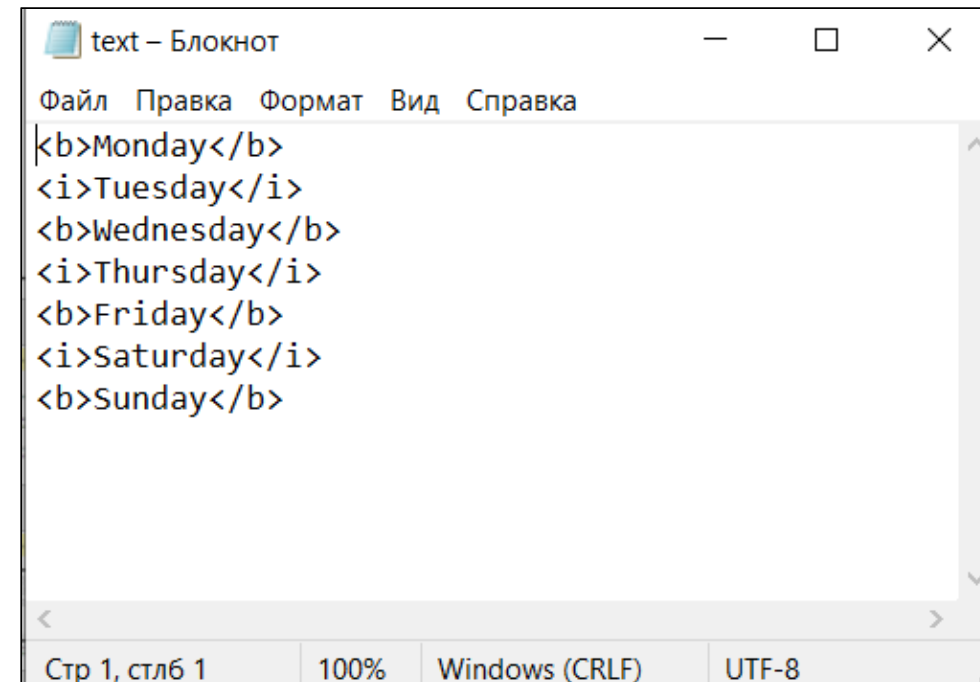
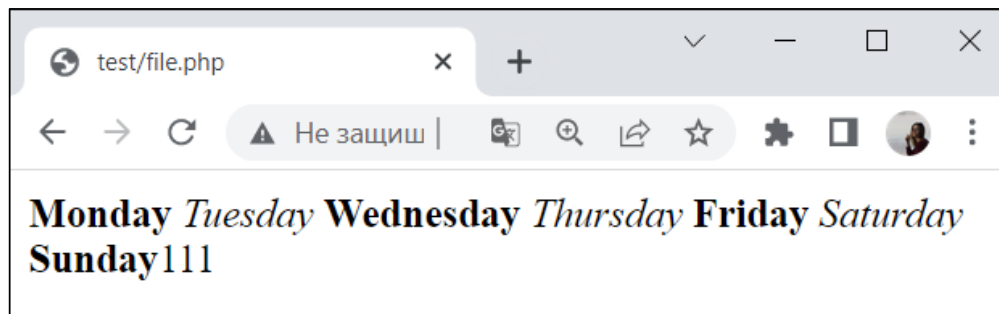
```
<?php  
echo readfile("text.txt");  
?>
```



Операции с данными файла

Также можно использовать функцию `fpassthru()` которая принимает один обязательный параметр – переменную, хранящую открытый файл. Т.е. перед использованием этой функции необходимо открыть файл в режиме чтения. По окончании считывания файла функция автоматически закрывает файл (при этом дескриптор файла становится недействительным).

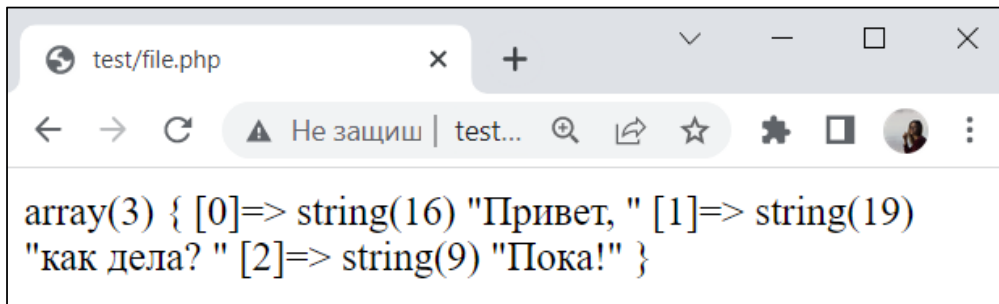
```
<?php
// Открываем файл в режиме чтения
$fp = fopen("text.txt", "r");
if ($fp) echo fpassthru($fp);
else echo "Ошибка при открытии файла";
?>
```



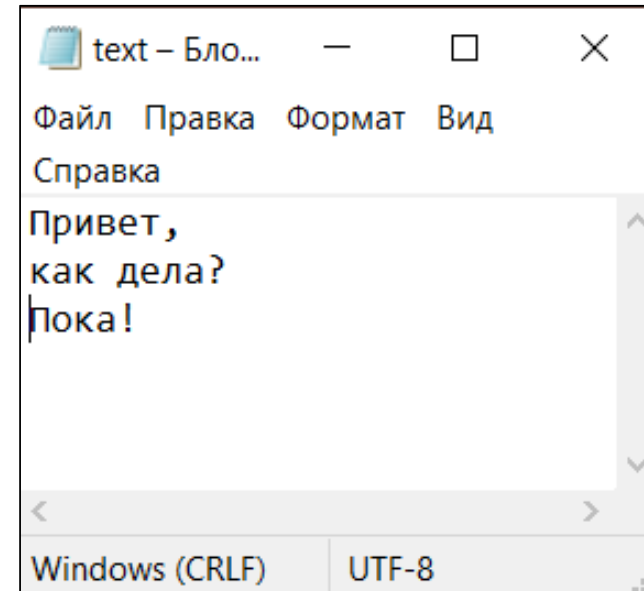
Операции с данными файла

Для считывания содержимое файла в массив используется функция `file()`. При вызове этой функции каждая строка файла сохраняется в отдельном элементе указанного массива. Однако не следует применять функцию `file()` к двоичным файлам (binary-safe), т.к. она не является безопасной в плане считывания двоичных файлов, если при этом где-то встретится символ конца файла (EOF), то она не гарантирует вам чтение всего двоичного файла.

```
<?php
$file_array = file("text.txt");
var_dump($file_array);
?>
```



```
array(3) { [0]=> string(16) "Привет, " [1]=> string(19)
"как дела? " [2]=> string(9) "Пока!" }
```



text – Бло... — □ ×

Файл Правка Формат Вид
Справка

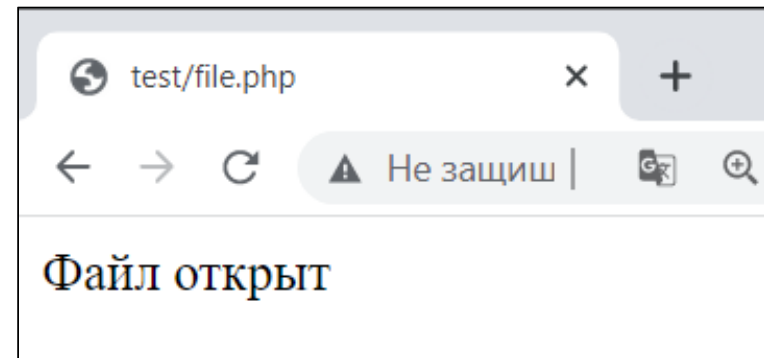
Привет,
как дела?
Пока!

Windows (CRLF) UTF-8

Операции с данными файла

Заккрытие файла происходит с помощью функции ***fclose()***, которая принимает один обязательный параметр – переменную с открытым файлом.

```
<?php
$fp = fopen("text.txt", "r");
if ($fp)
{
    echo 'Файл открыт';
    fclose($fp); // Заккрытие файла
}
?>
```



Операции с данными файла

```
fseek(resource $stream, int $offset, int $whence = SEEK_SET): int
```

Функция **fseek** устанавливает указатель файла **\$stream**, заданного в ее параметрах, на байт со смещением **\$offset** (от начала файла, от его конца или от текущей позиции, в зависимости от параметра **\$whence**).

Параметр **\$whence** задает, с какого места отсчитывается смещение **\$offset**.

Для расчета относительного смещения с помощью одного из следующих значений можно задать третий необязательный аргумент **\$whence** (точка отсчета):

SEEK_SET: начало файла плюс смещение.

SEEK_CUR (используется по умолчанию): текущая позиция плюс смещение.

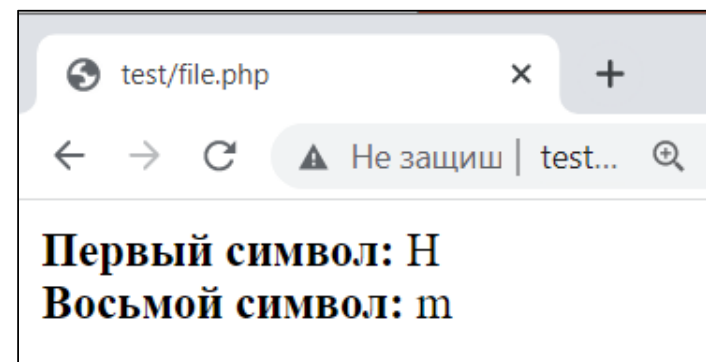
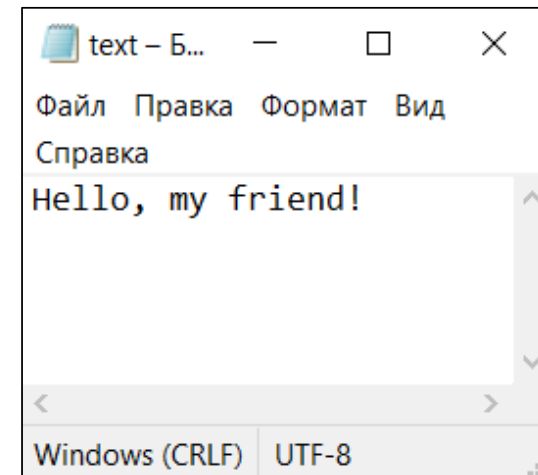
SEEK_END: конец файла плюс смещение.

В случае использования последних двух констант параметр **\$offset** вполне может быть отрицательным (а при применении **SEEK_END** он будет отрицательным). При этом функция **fseek** является целочисленной PHP функцией и возвращает 0, а не 1, в случае успеха, и -1 в случае неудачи.

Операции с данными файла

`fseek(resource $stream, int $offset, int $whence = SEEK_SET): int`

```
<?php
if (!$fp=fopen("text.txt","r"))
echo "Ошибка";
$one_char = fgetc($fp);
echo "<b>Первый символ: </b> $one_char";
fseek($fp, 7);
$one_char = fgetc($fp);
echo "<b>Восьмой символ:</b> $one_char";
fclose($fp);
?>
```

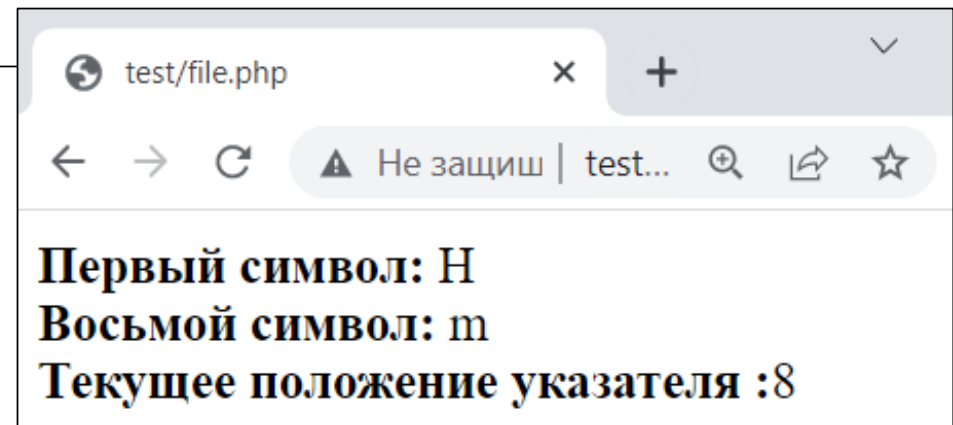


Операции с данными файла

`ftell(resource $stream): int`

Возвращает текущее положение указателя файла.

```
<?php
if (!$fp=fopen("text.txt","r"))
echo "Ошибка";
$one_char = fgetc($fp);
echo "<b>Первый символ: </b> $one_char";
fseek($fp, 7);
$one_char = fgetc($fp);
echo "<b>Восьмой символ:</b> $one_char";
echo "<B>Текущее положение указателя :</B>";
echo ftell($fp);
?>
```



Операции с данными файла

`rewind(resource $stream): bool`

Совместно с функцией `ftell` можно использовать функцию `rewind`. Данная функция принимает дескриптор файла и переустанавливает указатель позиции в начало этого файла.

При этом вызов

```
rewind($fp);
```

функционально аналогичен вызову

```
fseek($fp, 0);
```

Операции с данными файла

`rewind(resource $stream): bool`

```
<?php
if (!$fp=fopen("text.txt","r"))
echo "Ошибка";
$one_char = fgetc($fp);
echo "<b>Первый символ: </b> $one_char";
fseek($fp, 7);
$one_char = fgetc($fp);
echo "<b>Восьмой символ:</b> $one_char";
echo "<B>Текущее положение указателя :</B>";
echo ftell($fp);
rewind($fp);
echo "<B>Текущее положение указателя :</B>";
echo ftell($fp);
?>
```

