ПРАКТИЧЕСКАЯ РАБОТА №16

«Работа с файловой системой»

### Чтение и запись файлов

**Открытие и закрытие файлов**

Для открытия файлов в PHP определена функция **fopen()**. Она имеет следующее определение: resource fopen(string $filename, string $mode). Первый параметр $filename представляет путь к файлу, а второй - режим открытия. В зависимости от цели открытия и типа файла данный параметр может принимать следующие значения:

* 'r': файл открывается только для чтения. Если файла не существует, возвращает false
* 'r+': файл открывается только для чтения с возможностью записи. Если файла не существует, возвращает false
* 'w': файл открывается для записи. Если такой файл уже существует, то он перезаписывается, если нет - то он создается
* 'w+': файл открывается для записи с возможностью чтения. Если такой файл уже существует, то он перезаписывается, если нет - то он создается
* 'a': файл открывается для записи. Если такой файл уже существует, то данные записываются в конец файла, а старые данные остаются. Если файл не существует, то он создается
* 'a+': файл открывается для чтения и записи. Если файл уже существует, то данные дозаписываются в конец файла. Если файла нет, то он создается

Результатом функции fopen будет дескриптор файла. Этот дескриптор используется для операций с файлом и для его закрытия.

После окончания работы файл надо закрыть с помощью функции **fclose()**, которая принимает в качестве параметра дескриптор файла. Например, откроем и закроем файл:



Конструкция or die("текст ошибки") позволяет прекратить работу скрипта и вывесте некоторое сообщение об ошибке, если функция fopen не смогла открыть файл.

**Чтение файла**

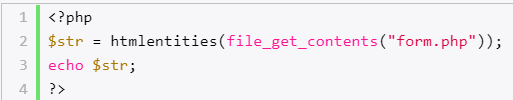
Для чтения файла можно использовать несколько функций. Для построчного чтения используется функция **fgets()**, которая получает дескриптор файла и возвращает одну считанную строку. Пройдем построчно по всему файлу:



При каждом вызове fgets() PHP будет помещать указатель в конец считанной строки. Чтобы проследить окончание файла, используется функция **feof()**, которая возвращает true при завершении файла. И пока не будет достигнут конец файла, мы можем применять функцию fgets().

**Чтение файла полностью**

Если нам надо прочитать файл полностью, то мы можем облегчить себе жизнь, применив функцию **file\_get\_contents()**:



При этом нам не надо открывать явно файл, получать дескриптор, а затем закрывать файл.

**Поблочное считывание**

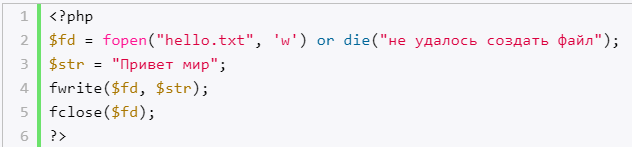
Также можно провести поблочное считывание, то есть считывать определенное количество байт из файла с помощью функции **fread()**:



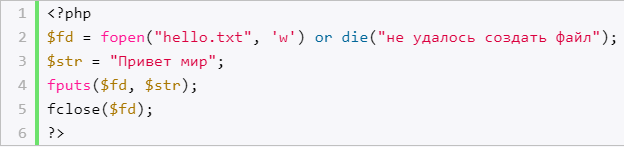
Функция fread() принимает два параметра: дескриптор считываемого файла и количество считываемых байтов. При считывании блока указатель в файле становится в конец этого блока. И также с помощью функции feof() можно отследить завершение файла.

**Запись файла**

Для записи файла применяется функция **fwrite()**, которая записывает в файл строку:



Аналогично работает другая функция **fputs()**:



**Работа с указателем файла**

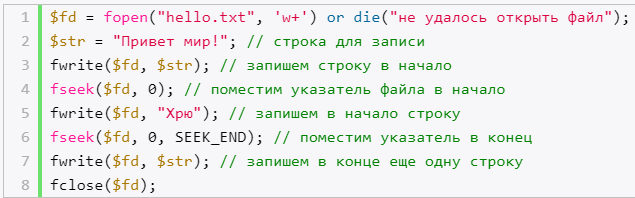
При открытии файла для чтения или записи в режиме 'w', указатель в файле помещается в начало. При считывании данных PHP перемещает указатель в файле в конец блока считанных данных. Однако мы также вручную можем управлять указателем в файле и устанавливать его в произвольное место. Для этого надо использовать функцию **fseek**, которая имеет следующее формальное определение:



Параметр $handle представляет дескриптор файла. Параметр $offset - смещение в байтах относительно начала файла, с которого начнется считывание/запись. Третий необязательный параметр задает способ установки смещения. Он может принимать три значения:

* **SEEK\_SET**: значение по умолчанию, устанавливает смещение в offset байт относительно начала файла
* **SEEK\_CUR**: устанавливает смещение в offset байт относительно начала текущей позиции в файле
* **SEEK\_END**: устанавливает смещение в offset байт от конца файла
* В случае удачной установки указателя функция fseek() возвращает 0, а при неудачной установке возвращает -1.

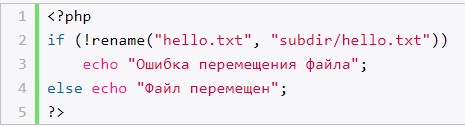
Пример использования функции:



### Управление файлами и каталогами

**Перемещение файла**

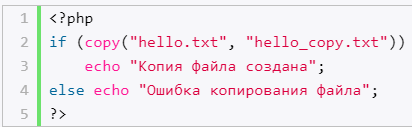
Для перемещения файла применяется функция **rename()**:



Если у нас в каталоге файла *hello.txt* имеется подкаталог *subdir*, то файл будет в него перемещен. Если файл был успешно перемещен, функция возвратит значение true.

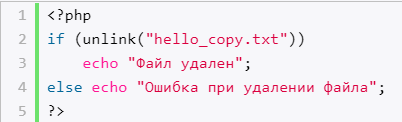
**Копирование файла**

Для копирования файла используется функция **copy()**. Она принимает имя копируемого файла, и имя копии файла. И если копирование прошло успешно, возвращает значение true:



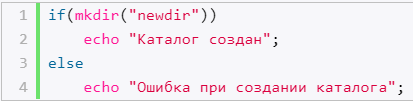
**Удаление файла**

Для удаления файла применяется функция **unlink**, которая принимает имя файла и возвращает значение true при успешном удалении файла:



**Создание каталога**

Для создания каталога используется функция **mkdir()**:

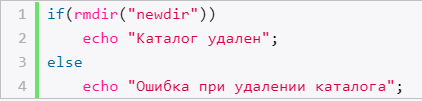


В данном случае mkdir создает новый каталог "newdir" в текущем каталоге. Если создание пройдет успешно, то функция возвращает значение true, иначе - false

Для создания новой папки в корневом каталоге можно использовать выражение mkdir("/newdir").

**Удаление каталога**

Для удаления каталога применяется функция **rmdir()**. Ее использование аналогично mkdir():



**Операции с каталогами**

Для получения абсолютного пути к текущему каталогу используется функция **getcwd()**, которая возвращает путь в виде строки:



Функция **opendir()** открывает определенный каталог для считывания из него информации о файлах и каталогах. При успешном открытии каталога функция возвращает дескриптор открытого каталога. После окончания работы с каталогом его надо закрыть функцией **closedir()**.

Для считывания имени отдельного файла в открытом каталоге применяется функция **readdir()**.

Теперь объединим эти функции и выведем на страницу все файлы и подкаталоги из текущего каталога:



### Блокировка файлов. Функция flock

Чтобы ограничить доступ к файлу, в PHP используется функция **flock()**. Эта функция блокирует доступ к файлу, когда он уже занят одним пользователем, а все остальные запросы ставит в очередь. При освобождении файла он разблокируется, передается первому в очереди пользователю и снова блокируется.

Функция имеет следующее определение:



Первый параметр - дескриптор файла, возвращаемые функцией fopen().

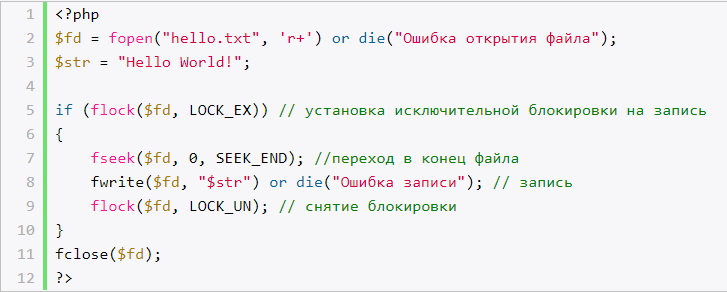
Второй параметр указывает на тип блокировки. Он может принимать следующие значения:

* **LOCK\_SH** (или число 1): разделяемая блокировка (чтение файла)
* **LOCK\_EX** (или число 2): исключительная блокировка (запись файла)
* **LOCK\_UN** (или число 3): для снятия блокировки
* **LOCK\_NB** (или число 4): эта константа используется только вместе с одной из предыдущих в битовой маске (LOCK\_EX | LOCK\_NB), если не надо ждать пока flock() получит блокировку.

Третий необязательный параметр $wouldblock будет содержать true, если блокировка будет блокирующей.

При успешном выполнении функция flock возвратит значение true, а в случае ошибки - false.

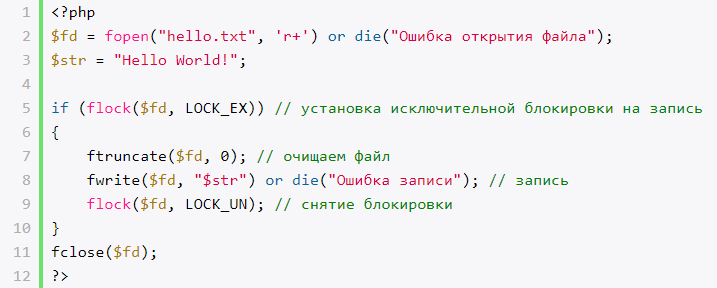
Используем flock для ограничения доступа к файлу:



При изменении файла блокировка ставится непосредственно перед внесением изменений, а снимается сразу после их внесения. Иначе программа может замедлить свою работу. Поэтому вызов, блокирующий файл: flock($fd, LOCK\_EX) поставлен непосредственно перед вызовом функции fwrite($fd, "$str"). А снятие блокировки с помощью константы LOCK\_UN идет сразу после записи.

Обратите внимание, что при открытии файла здесь использовался режим 'r+', а не 'w' или 'w+', так как 'w' и 'w+' уже подразумевают изменение файла. Поэтому при блокировке, даже если надо делать запись, не рекомендуется использование 'w' и 'w+'.

Если нам надо стереть все содержимое файла и перезаписать файл по-новому, то мы можем воспользоваться функцией **ftruncate**:



### Отправка файлов на сервер

Чтобы загрузить файл на сервер, нам надо использовать форму с параметром enctype='multipart/form-data' и массив $\_FILES. Итак, создадим файл *upload.php* со следующим содержимым:



Здесь определена форм с атрибутом enctype='multipart/form-data'. Форма содержит специальное поле для выбора файла.

Все загружаемые файлы попадают в ассоциативный массив **$\_FILES**. Чтобы определить, а есть ли вообще загруженные файлы, можно использовать конструкцию if: if ($\_FILES)

Массив $\_FILES является двухмерным. Мы можем загрузить набор файлов, и каждый загруженный файл можно получить по ключу, который совпадает со значением атрибута name.

Так как элемент для загрузки файла на форме имеет name='filename', то данный файл мы можем получить с помощью выражения $\_FILES['filename'].

У каждого объекта файла есть свои параметры, которые мы можем получить:

* $\_FILES['file']['name']: имя файла
* $\_FILES['file']['type']: тип содержимого файла, например, image/jpeg
* $\_FILES['file']['size']: размер файла в байтах
* $\_FILES['file']['tmp\_name']: имя временного файла, сохраненного на сервере
* $\_FILES['file']['error']: код ошибки при загрузке
* Также мы можем проверить наличие ошибок при загрузке. Если у нас нет ошибки, то поле $\_FILES['filename']['error'] содержит значение UPLOAD\_ERR\_OK.

При отправке файла на сервер он сначала загружается во временное место, из которого затем с помощью функции **move\_uploaded\_file()** он перемещается в каталог сервера.

Функция move\_uploaded\_file() принимает два параметра путь к загруженному временному файлу и путь, куда надо поместить загруженный файл.

**Ограничения и настройка загрузки**

По умолчанию размер загружаемых файлов ограничен 2 мб. Однако можно настроить данный показатель в файле конфигурации. Изменим этот показатель, например, до 10 мб. Для этого найдем в файле *php.ini* следующую строку:



Изменим ее на



Также мы можем настроить папку для временных загружаемых файлов. Для этого в файле *php.ini* найдем следующую строку:



Изменим ее на

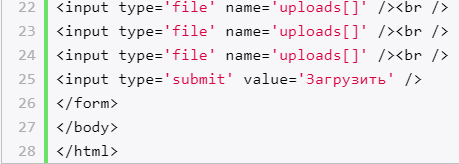


Также в каталоге php нам надо создать папку **upload**.

**Мультизагрузка**

Изменим скрипт *upload.php* так, чтобы он поддерживал множественную загрузку:





Каждое поле выбора файла имеет атрибут name='uploads[]', поэтому сервер будет рассматривать набор отправленных файлов как единый массив.

Затем используя цикл foreach, проходим по все файлам и сохраняем их в каталог веб-сайта.