http://php.net/manual/en/function.fread.php

**Description** [**¶**](http://php.net/manual/en/function.fread.php#refsect1-function.fread-description)

string **fread** ( resource $handle , int $length )

**fread()** reads up to length bytes from the file pointer referenced by handle. Reading stops as soon as one of the following conditions is met:

* length bytes have been read
* EOF (end of file) is reached
* a packet becomes available or the [socket timeout](http://php.net/manual/en/function.socket-set-timeout.php) occurs (for network streams)
* if the stream is read buffered and it does not represent a plain file, at most one read of up to a number of bytes equal to the chunk size (usually 8192) is made; depending on the previously buffered data, the size of the returned data may be larger than the chunk size.

**Parameters** [**¶**](http://php.net/manual/en/function.fread.php#refsect1-function.fread-parameters)

handle

A file system pointer [resource](http://php.net/manual/en/language.types.resource.php) that is typically created using [fopen()](http://php.net/manual/en/function.fopen.php).

length

Up to length number of bytes read.

**Return Values** [**¶**](http://php.net/manual/en/function.fread.php#refsect1-function.fread-returnvalues)

Returns the read string or **FALSE** on failure.

**Examples** [**¶**](http://php.net/manual/en/function.fread.php#refsect1-function.fread-examples)

**Example #1 A simple fread() example**

<?php
// get contents of a file into a string
$filename = "/usr/local/something.txt";
$handle = fopen($filename, "r");
$contents = fread($handle, filesize($filename));
fclose($handle);
?>

**Example #2 Binary fread() example**

**Warning**

On systems which differentiate between binary and text files (i.e. Windows) the file must be opened with 'b' included in [fopen()](http://php.net/manual/en/function.fopen.php) mode parameter.

<?php
$filename = "c:\\files\\somepic.gif";
$handle = fopen($filename, "rb");
$contents = fread($handle, filesize($filename));
fclose($handle);
?>

**Example #3 Remote fread() examples**

**Warning**

When reading from anything that is not a regular local file, such as streams returned when reading [remote files](http://php.net/manual/en/features.remote-files.php) or from [popen()](http://php.net/manual/en/function.popen.php) and [fsockopen()](http://php.net/manual/en/function.fsockopen.php), reading will stop after a packet is available. This means that you should collect the data together in chunks as shown in the examples below.

<?php
// For PHP 5 and up
$handle = fopen("http://www.example.com/", "rb");
$contents = stream\_get\_contents($handle);
fclose($handle);
?>

<?php
$handle = fopen("http://www.example.com/", "rb");
if (FALSE === $handle) {
    exit("Failed to open stream to URL");
}

$contents = '';

while (!feof($handle)) {
    $contents .= fread($handle, 8192);
}
fclose($handle);
?>

**Notes** [**¶**](http://php.net/manual/en/function.fread.php#refsect1-function.fread-notes)

**Note**:

If you just want to get the contents of a file into a string, use [file\_get\_contents()](http://php.net/manual/en/function.file-get-contents.php) as it has much better performance than the code above.

**Note**:

Note that **fread()** reads from the current position of the file pointer. Use [ftell()](http://php.net/manual/en/function.ftell.php) to find the current position of the pointer and [rewind()](http://php.net/manual/en/function.rewind.php) to rewind the pointer position.

**See Also** [**¶**](http://php.net/manual/en/function.fread.php#refsect1-function.fread-seealso)

* [fwrite()](http://php.net/manual/en/function.fwrite.php) - Binary-safe file write
* [fopen()](http://php.net/manual/en/function.fopen.php) - Opens file or URL
* [fsockopen()](http://php.net/manual/en/function.fsockopen.php) - Open Internet or Unix domain socket connection
* [popen()](http://php.net/manual/en/function.popen.php) - Opens process file pointer
* [fgets()](http://php.net/manual/en/function.fgets.php) - Gets line from file pointer
* [fgetss()](http://php.net/manual/en/function.fgetss.php) - Gets line from file pointer and strip HTML tags
* [fscanf()](http://php.net/manual/en/function.fscanf.php) - Parses input from a file according to a format
* [file()](http://php.net/manual/en/function.file.php) - Reads entire file into an array
* [fpassthru()](http://php.net/manual/en/function.fpassthru.php) - Output all remaining data on a file pointer
* [ftell()](http://php.net/manual/en/function.ftell.php) - Returns the current position of the file read/write pointer
* [rewind()](http://php.net/manual/en/function.rewind.php) - Rewind the position of a file pointer

[add a note](http://php.net/manual/add-note.php?sect=function.fread&redirect=http://php.net/manual/en/function.fread.php)

**User Contributed Notes 41 notes**

[up](http://php.net/manual/vote-note.php?id=93250&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=93250&page=function.fread&vote=down)

20

[***edgarinvillegas at hotmail dot com*** ¶](http://php.net/manual/en/function.fread.php#93250)

**6 years ago**

I had a fread script that hanged forever (from php manual):

<?php
$fp = fsockopen("example.host.com", 80);
if (!$fp) {
    echo "$errstr ($errno)<br />\n";
} else {
    fwrite($fp, "Data sent by socket");
    $content = "";
    while (!feof($fp)) {  //This looped forever
        $content .= fread($fp, 1024);
    }
    fclose($fp);
    echo $content;
}
?>

The problem is that sometimes end of streaming is not marked by EOF nor a fixed mark, that's why this looped forever. This caused me a lot of headaches...
I solved it using the stream\_get\_meta\_data function and a break statement as the following shows:

<?php
$fp = fsockopen("example.host.com", 80);
if (!$fp) {
    echo "$errstr ($errno)<br />\n";
} else {
    fwrite($fp, "Data sent by socket");
    $content = "";
    while (!feof($fp)) {
        $content .= fread($fp, 1024);
        $stream\_meta\_data = stream\_get\_meta\_data($fp); //Added line
         if($stream\_meta\_data['unread\_bytes'] <= 0) break; //Added line
    }
    fclose($fp);
    echo $content;
}
?>

Hope this will save a lot of headaches to someone.

(Greetings, from La Paz-Bolivia)

[up](http://php.net/manual/vote-note.php?id=113885&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=113885&page=function.fread&vote=down)

10

[***dharmilshah at gmail dot com*** ¶](http://php.net/manual/en/function.fread.php#113885)

**2 years ago**

For anyone still trying to write an effective file downloader function/script, the work has been done for you in all the major servers including Apache & nginx.

Using the X-Sendfile header, you can do the following:

if ($user->isLoggedIn())
{
    header("X-Sendfile: $path\_to\_somefile\_private");
    header("Content-Type: application/octet-stream");
    header("Content-Disposition: attachment; filename=\"$somefile\"");
}

Apache will serve the file for you while NOT revealing your private file path! Pretty nice. This works on all browsers/download managers and saves a lot of resources.

Documentation:
Apache module: <https://tn123.org/mod_xsendfile/>
Nginx: <http://wiki.nginx.org/XSendfile>
Lighttpd: <http://blog.lighttpd.net/articles/2006/07/02/x-sendfile/>

Hopefully this will save you many hours of work.

[up](http://php.net/manual/vote-note.php?id=84115&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=84115&page=function.fread&vote=down)

14

[***Edward Jaramilla*** ¶](http://php.net/manual/en/function.fread.php#84115)

**7 years ago**

I couldn't get some of the previous resume scripts to work with Free Download Manager or Firefox.  I did some clean up and modified the code a little.

Changes:
1. Added a Flag to specify if you want download to be resumable or not
2. Some error checking and data cleanup for invalid/multiple ranges based on <http://tools.ietf.org/id/draft-ietf-http-range-retrieval-00.txt>
3. Always calculate a $seek\_end even though the range specification says it could be empty... eg: bytes 500-/1234
4. Removed some Cache headers that didn't seem to be needed. (add back if you have problems)
5. Only send partial content header if downloading a piece of the file (IE workaround)

<?php

function dl\_file\_resumable($file, $is\_resume=TRUE)
{
    //First, see if the file exists
    if (!is\_file($file))
    {
        die("<b>404 File not found!</b>");
    }

    //Gather relevent info about file
    $size = filesize($file);
    $fileinfo = pathinfo($file);

    //workaround for IE filename bug with multiple periods / multiple dots in filename
    //that adds square brackets to filename - eg. setup.abc.exe becomes setup[1].abc.exe
    $filename = (strstr($\_SERVER['HTTP\_USER\_AGENT'], 'MSIE')) ?
                  preg\_replace('/\./', '%2e', $fileinfo['basename'], substr\_count($fileinfo['basename'], '.') - 1) :
                  $fileinfo['basename'];

    $file\_extension = strtolower($path\_info['extension']);

    //This will set the Content-Type to the appropriate setting for the file
    switch($file\_extension)
    {
        case 'exe': $ctype='application/octet-stream'; break;
        case 'zip': $ctype='application/zip'; break;
        case 'mp3': $ctype='audio/mpeg'; break;
        case 'mpg': $ctype='video/mpeg'; break;
        case 'avi': $ctype='video/x-msvideo'; break;
        default:    $ctype='application/force-download';
    }

    //check if http\_range is sent by browser (or download manager)
    if($is\_resume && isset($\_SERVER['HTTP\_RANGE']))
    {
        list($size\_unit, $range\_orig) = explode('=', $\_SERVER['HTTP\_RANGE'], 2);

        if ($size\_unit == 'bytes')
        {
            //multiple ranges could be specified at the same time, but for simplicity only serve the first range
            //<http://tools.ietf.org/id/draft-ietf-http-range-retrieval-00.txt>
            list($range, $extra\_ranges) = explode(',', $range\_orig, 2);
        }
        else
        {
            $range = '';
        }
    }
    else
    {
        $range = '';
    }

    //figure out download piece from range (if set)
    list($seek\_start, $seek\_end) = explode('-', $range, 2);

    //set start and end based on range (if set), else set defaults
    //also check for invalid ranges.
    $seek\_end = (empty($seek\_end)) ? ($size - 1) : min(abs(intval($seek\_end)),($size - 1));
    $seek\_start = (empty($seek\_start) || $seek\_end < abs(intval($seek\_start))) ? 0 : max(abs(intval($seek\_start)),0);

    //add headers if resumable
    if ($is\_resume)
    {
        //Only send partial content header if downloading a piece of the file (IE workaround)
        if ($seek\_start > 0 || $seek\_end < ($size - 1))
        {
            header('HTTP/1.1 206 Partial Content');
        }

        header('Accept-Ranges: bytes');
        header('Content-Range: bytes '.$seek\_start.'-'.$seek\_end.'/'.$size);
    }

    //headers for IE Bugs (is this necessary?)
    //header("Cache-Control: cache, must-revalidate");
    //header("Pragma: public");

    header('Content-Type: ' . $ctype);
    header('Content-Disposition: attachment; filename="' . $filename . '"');
    header('Content-Length: '.($seek\_end - $seek\_start + 1));

    //open the file
    $fp = fopen($file, 'rb');
    //seek to start of missing part
    fseek($fp, $seek\_start);

    //start buffered download
    while(!feof($fp))
    {
        //reset time limit for big files
        set\_time\_limit(0);
        print(fread($fp, 1024\*8));
        flush();
        ob\_flush();
    }

    fclose($fp);
    exit;
}

?>

[up](http://php.net/manual/vote-note.php?id=82841&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=82841&page=function.fread&vote=down)

6

[***mail at 3v1n0 dot net*** ¶](http://php.net/manual/en/function.fread.php#82841)

**7 years ago**

This is an hack I've done to download remote files with HTTP resume support. This is useful if you want to write a download script that fetches files remotely and then sends them to the user, adding support to download managers (I tested it on wget). To do that you should also use a "remote\_filesize" function that you can easily write/find.

<?php
function readfile\_chunked\_remote($filename, $seek = 0, $retbytes = true, $timeout = 3) {
    set\_time\_limit(0);
    $defaultchunksize = 1024\*1024;
    $chunksize = $defaultchunksize;
    $buffer = '';
    $cnt = 0;
    $remotereadfile = false;

    if (preg\_match('/[a-zA-Z]+:\/\//', $filename))
        $remotereadfile = true;

    $handle = @fopen($filename, 'rb');

    if ($handle === false) {
        return false;
    }

    stream\_set\_timeout($handle, $timeout);

    if ($seek != 0 && !$remotereadfile)
        fseek($handle, $seek);

    while (!feof($handle)) {

        if ($remotereadfile && $seek != 0 && $cnt+$chunksize > $seek)
            $chunksize = $seek-$cnt;
        else
            $chunksize = $defaultchunksize;

        $buffer = @fread($handle, $chunksize);

        if ($retbytes || ($remotereadfile && $seek != 0)) {
            $cnt += strlen($buffer);
        }

        if (!$remotereadfile || ($remotereadfile && $cnt > $seek))
            echo $buffer;

        ob\_flush();
        flush();
    }

    $info = stream\_get\_meta\_data($handle);

    $status = fclose($handle);

    if ($info['timed\_out'])
        return false;

    if ($retbytes && $status) {
        return $cnt;
    }

    return $status;
}
?>

[up](http://php.net/manual/vote-note.php?id=47832&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=47832&page=function.fread&vote=down)

3

[***ibis at connect dot ie*** ¶](http://php.net/manual/en/function.fread.php#47832)

**11 years ago**

If, like me, you're in the habit of using fopen("<http://...>") and fread for pulling fairly large remote files, you may find that the upgrade to PHP5 (5.0.2 on Win2000/IIS5) causes fread to top out at about 8035 bytes. PHP5 RC2 with identical php.ini settings did not exhibit this behaviour (I was using this for testing). Irritating for me because I was using simple\_xml\_load to load the file contents as XML, and the problem initially appeared to be that function.

Solution - swap over to file\_get\_contents or use the loop suggested on the documentation above (see Warning).

[up](http://php.net/manual/vote-note.php?id=50904&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=50904&page=function.fread&vote=down)

2

[***dvsoftware at gmail dot com*** ¶](http://php.net/manual/en/function.fread.php#50904)

**10 years ago**

I was trying to implement resume support in download script, and i have finnaly succeded. here is the script:

<?php
function dl\_file\_resume($file){

   //First, see if the file exists
   if (!is\_file($file)) { die("<b>404 File not found!</b>"); }

   //Gather relevent info about file
   $len = filesize($file);
   $filename = basename($file);
   $file\_extension = strtolower(substr(strrchr($filename,"."),1));

   //This will set the Content-Type to the appropriate setting for the file
   switch( $file\_extension ) {
     case "exe": $ctype="application/octet-stream"; break;
     case "zip": $ctype="application/zip"; break;
     case "mp3": $ctype="audio/mpeg"; break;
     case "mpg":$ctype="video/mpeg"; break;
     case "avi": $ctype="video/x-msvideo"; break;

     //The following are for extensions that shouldn't be downloaded (sensitive stuff, like php files)
     case "php":
     case "htm":
     case "html":
     case "txt": die("<b>Cannot be used for ". $file\_extension ." files!</b>"); break;

     default: $ctype="application/force-download";
   }

   //Begin writing headers
   header("Pragma: public");
   header("Expires: 0");
   header("Cache-Control:");
   header("Cache-Control: public");
   header("Content-Description: File Transfer");

   //Use the switch-generated Content-Type
   header("Content-Type: $ctype");
$filespaces = str\_replace("\_", " ", $filename);

//if your filename contains underscores, you can replace them with spaces
  $header='Content-Disposition: attachment; filename='.$filespaces.';';
   header($header );
   header("Content-Transfer-Encoding: binary");

  $size=filesize($file);
//check if http\_range is sent by browser (or download manager)
   if(isset($\_ENV['HTTP\_RANGE'])) {
list($a, $range)=explode("=",$\_ENV['HTTP\_RANGE']);
//if yes, download missing part
str\_replace($range, "-", $range);
$size2=$size-1;
header("Content-Range: $range$size2/$size");
$new\_length=$size2-$range;
header("Content-Length: $new\_length");
/if not, download whole file
} else {
$size2=$size-1;
header("Content-Range: bytes 0-$size2/$size");
header("Content-Length: ".$size2);
}
//open the file
$fp=fopen("$file","r");
//seek to start of missing part
fseek($fp,$range);
//start buffered download
while(!feof($fp))
{
//reset time limit for big files
set\_time\_limit();
print(fread($fp,1024\*8));
flush();
}
fclose($fp);

   exit;

}
?>

EXAMPLE
<?php
dl\_file\_resume("somefile.mp3");
?>

please write if you find any errors, i have tested this only with mp3 files, but others should be fine

[up](http://php.net/manual/vote-note.php?id=105471&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=105471&page=function.fread&vote=down)

3

[***randym*** ¶](http://php.net/manual/en/function.fread.php#105471)

**4 years ago**

Concerning [problems with UTF-8 and] downloading Zip files I found that simply adding 3 lines of code before starting the fread to the buffer for delivery in all browsers solved the problem.

<?php
    ob\_end\_clean();
    ob\_start();
    header( 'Content-Type:' );
?>

... see where placed in the function below:

<?php
function readfile\_chunked( $filename, $retbytes = true ) {
    $chunksize = 1 \* (1024 \* 1024); // how many bytes per chunk
    $buffer = '';
    $cnt = 0;
    $handle = fopen( $filename, 'rb' );
    if ( $handle === false ) {
        return false;
    }
    ob\_end\_clean(); //added to fix ZIP file corruption
    ob\_start(); //added to fix ZIP file corruption
    header( 'Content-Type:' ); //added to fix ZIP file corruption
    while ( !feof( $handle ) ) {
        $buffer = fread( $handle, $chunksize );
        //$buffer = str\_replace("ï»¿","",$buffer);
        echo $buffer;
        ob\_flush();
        flush();
        if ( $retbytes ) {
            $cnt += strlen( $buffer );
        }
    }
    $status = fclose( $handle );
    if ( $retbytes && $status ) {
        return $cnt; // return num. bytes delivered like readfile() does.
    }
    return $status;
}
?>

[up](http://php.net/manual/vote-note.php?id=42381&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=42381&page=function.fread&vote=down)

3

[***webmaster at wildpeaks dot com*** ¶](http://php.net/manual/en/function.fread.php#42381)

**11 years ago**

The following function retrieves a line in a file, regardless of its size, so you won't get an error if the file's size is beyond php's allowed memory limit (the string has to be below however), which is something i was needing for accessing a big log file generated by a webhost. Indexes start at 1 (so $line = 1 means the first line unlike arrays). If the file is small, it would be better to use "file()" however.

<?php
function strpos\_count($haystack, $needle, $i = 0) {
    while (strpos($haystack,$needle) !== false) {$haystack = substr($haystack, (strpos($haystack,$needle) + 1)); $i++;}
    return $i;
}
function getLine($file,$line=1){
    $occurence = 0;
    $contents = '';
    $startPos = -1;
    if (!file\_exists($file)) return '';
    $fp = @fopen($file, "rb");
    if (!$fp) return '';
    while (!@feof($fp)) {
        $str = @fread($fp, 1024);
        $number\_of\_occurences = strpos\_count($str,"\n");
        if ($number\_of\_occurences == 0) {if ($start\_pos != -1) {$contents .= $str;}}
        else {
            $lastPos = 0;
            for ($i = 0; $i < $number\_of\_occurences; $i++){
                $pos = strpos($str,"\n", $lastPos);
                $occurence++;
                if ($occurence == $line) {
                    $startPos = $pos;
                    if ($i == $number\_of\_occurences - 1) {$contents = substr($str, $startPos + 1);}
                } elseif ($occurence == $line + 1) {
                    if ($i == 0) {$contents .= substr($str, 0, $pos);} else {$contents = substr($str, $startPos, $pos - $startPos);}
                    $occurence = 0;
                    break;
                }
                $lastPos = $pos + 1;
            }
        }
    }
    @fclose($fp);
    return $contents;
}
?>

[up](http://php.net/manual/vote-note.php?id=49078&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=49078&page=function.fread&vote=down)

3

[***m (at) mindplay (dot) dk*** ¶](http://php.net/manual/en/function.fread.php#49078)

**10 years ago**

Here's a function for sending a file to the client - it may look more complicated than necessary, but has a number of advantages over simpler file sending functions:

- Works with large files, and uses only an 8KB buffer per transfer.

- Stops transferring if the client is disconnected (unlike many scripts, that continue to read and buffer the entire file, wasting valuable resources) but does not halt the script

- Returns TRUE if transfer was completed, or FALSE if the client was disconnected before completing the download - you'll often need this, so you can log downloads correctly.

- Sends a number of headers, including ones that ensure it's cached for a maximum of 2 hours on any browser/proxy, and "Content-Length" which most people seem to forget.

(tested on Linux (Apache) and Windows (IIS5/6) under PHP4.3.x)

Note that the folder from which protected files will be pulled, is set as a constant in this function (/protected) ... Now here's the function:

<?php
function send\_file($name) {
  ob\_end\_clean();
  $path = "protected/".$name;
  if (!is\_file($path) or connection\_status()!=0) return(FALSE);
  header("Cache-Control: no-store, no-cache, must-revalidate");
  header("Cache-Control: post-check=0, pre-check=0", false);
  header("Pragma: no-cache");
  header("Expires: ".gmdate("D, d M Y H:i:s", mktime(date("H")+2, date("i"), date("s"), date("m"), date("d"), date("Y")))." GMT");
  header("Last-Modified: ".gmdate("D, d M Y H:i:s")." GMT");
  header("Content-Type: application/octet-stream");
  header("Content-Length: ".(string)(filesize($path)));
  header("Content-Disposition: inline; filename=$name");
  header("Content-Transfer-Encoding: binary\n");
  if ($file = fopen($path, 'rb')) {
    while(!feof($file) and (connection\_status()==0)) {
      print(fread($file, 1024\*8));
      flush();
    }
    fclose($file);
  }
  return((connection\_status()==0) and !connection\_aborted());
}
?>

And here's an example of using the function:

<?php
if (!send\_file("platinumdemo.zip")) {
die ("file transfer failed");
// either the file transfer was incomplete
// or the file was not found
} else {
// the download was a success
// log, or do whatever else
}
?>

Regards,
Rasmus Schultz

[up](http://php.net/manual/vote-note.php?id=95413&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=95413&page=function.fread&vote=down)

1

[***Tblue*** ¶](http://php.net/manual/en/function.fread.php#95413)

**5 years ago**

Note that fread() returns an empty string if you try to read beyond EOF, while the manual states otherwise ("Returns [...] FALSE on failure."). This e. g. happens with empty files (0 bytes long).

This does not look like a bug in PHP's fread() implementation to me, but rather like a documentation bug. The manpage for the C function fread() states:
> fread() does not distinguish between end-of-file and error, and callers must use feof(3) and ferror(3) to determine which occurred.
It also says:
> If an error occurs, or the end-of-file is reached, the return value is a short item count (or zero).

That means that in the case of empty files, C's fread() returns 0 and thus we get an empty PHP string: PHP's fread() does not seem to check for errors as the manpage recommends; that's fine, the PHP programmer has to do it, but it would be nice if this behaviour would be explicitly documented.

In short: When using fread(), you have to check for FALSE and empty strings. Consider the following wrapper function:
<?php
function my\_fread( $handle, $length )
{
  if( ( $ret = fread( $handle, $length ) ) === '' )
  {
    return false;
  }

  return $ret;
}
?>

[up](http://php.net/manual/vote-note.php?id=75063&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=75063&page=function.fread&vote=down)

1

[***kai at froghh dot de*** ¶](http://php.net/manual/en/function.fread.php#75063)

**8 years ago**

reading from a socket stream can be different to the
behaviour expected, since you have not set
stream\_set\_blocking to 1.
sample source:
<?php
$fp = fsockopen ($server, $port, $errno, $errstr, $socket\_timeout);
$header = '';
do {
    $header.=fread($fp,1);
    $i++;
} while (!preg\_match('/\\r\\n\\r\\n$/', $header) && $i < $maxheaderlenth);
preg\_match('/Content\\-Length:\\s+([0-9]\*)\\r\\n/', $header,$matches);
$buffer = fread($this->\_fp, $matches[1]);
?>

if i.e. the content length is 50000 and the responding server is to slow
(means 50000 are not completely sent when fread is called)
you'll only receive the number of bytes sent by the
responding server at the time fread is called.

fread will not wait for any data to complete the given size.
as described in user notes on stream\_set\_blocking there
seems to be a bug using stream\_set\_blocking.
a workaround - well, not the best way - is to read
the response split to 1 byte
instead of
<?php $buffer = fread($this->\_fp, $matches[1]); ?>

you'd write
<?php
$buffer = '';
for($i = 0; $i < $matches[1]; $i++){
    $buffer .= fread($this->\_fp, 1);
}
?>

it several tests this seems like it works.

[up](http://php.net/manual/vote-note.php?id=56991&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=56991&page=function.fread&vote=down)

1

[***adamgamble at gmail dot com*** ¶](http://php.net/manual/en/function.fread.php#56991)

**10 years ago**

<?php
/\*
geoCode($address)
Accepts an address in the form of
999 Geocode Dr. New York, Ny 10108
returns array with lat and lon
\*/

function geoCode($address) {

    $gaddress = "<http://maps.google.com?q=>" . urlencode($address);

    $handle = fopen($gaddress, "r");
    $contents = '';

     while (!feof($handle)) {
         $contents .= fread($handle, 8192);
     }
     fclose($handle);
     ereg('<center lat="([0-9.-]{1,})" lng="([0-9.-]{1,})"/>', $contents, $regs);

     $returnData['lat'] = $regs[1];
     $returnData['lon'] = $regs[2];

     return $returnData;
}

print\_r(geoCode("1064 Georgetown ln. Birmingham, Al 35217"));
?>

[up](http://php.net/manual/vote-note.php?id=50637&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=50637&page=function.fread&vote=down)

1

[***fenris\_wolf0 at yahoo dot com*** ¶](http://php.net/manual/en/function.fread.php#50637)

**10 years ago**

To make the effects of the latest PHP version changes of the fread function even more explicit:  the new size limitation of fread -regardless of the filesize one specifies,  in the example below 1024 \* 1024- means that if one was  simply reading the contents of a text file from a dynamic URL like so:

<?php
  $dp = "<http://www.example.com/filename.php>";
  $buffer = fopen($dp, 'r');
  if (!$buffer)
    {
      echo("<P>Error: unable to load URL file into $buffer.      Process  aborted.</P>");
      exit();
    }
  $sp = fread($buffer, 1024\*1024);
  fclose($buffer);
  highlight\_string($sp);
?>

one should from now on use the file\_get\_contents function, as shown below, to avoid one's text being truncated forcibly.

<?php
  $dp = "<http://www.example.com/filename.php>";
  if (!$dp)
    {
      echo("<P>Error: unable to load URL file into $dp.  Process aborted.</P>");
      exit();
    }
  $sp = file\_get\_contents($dp);
  highlight\_string($sp);
?>

I thought it couldn't hurt to clarify this detail in order to save time for anyone else who is in the same situation as I was tonight when my ISP abruptly upgraded to the latest version of PHP...    :(

Thank you to every previous contributor to this topic.

[up](http://php.net/manual/vote-note.php?id=46517&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=46517&page=function.fread&vote=down)

1

[***fpinho at hotpop dot com*** ¶](http://php.net/manual/en/function.fread.php#46517)

**11 years ago**

After using the suggested function from Rasmus Schultz : mindplay(at)mindplay(dot)dk, I've just noticed that people trying to download big files with a slow connection would get download stopped after exactly 60seconds -> the max execution time set with php.ini.
I suggest using a bigger buffer (1024x1024), or maybe resetting the time limit within the 'while' cicle with:
   set\_time\_limit(0);

The cicle would go like this:

<?php
   while(!feof($file) and (connection\_status()==0)) {
     print(fread($file, 1024\*1024));
     set\_time\_limit(0);
     flush();
   }
?>

Frederico Pinho

[up](http://php.net/manual/vote-note.php?id=82474&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=82474&page=function.fread&vote=down)

2

[***shocker at shockingsoft dot com*** ¶](http://php.net/manual/en/function.fread.php#82474)

**7 years ago**

If you read from a socket connection or any other stream that may delay when responsing but you want to set a timeout you can use stream\_set\_timeout():

<?php
$f = fsockopen("127.0.0.1", 123);
if ($f)
{
  fwrite($f, "hello");
  stream\_set\_timeout($f, 5); //5 seconds read timeout
  if (!fread($f, 5)) echo "Error while reading";
    else echo "Read ok";
  fclose($f);
}
?>

[up](http://php.net/manual/vote-note.php?id=102986&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=102986&page=function.fread&vote=down)

2

[***Anonymous*** ¶](http://php.net/manual/en/function.fread.php#102986)

**4 years ago**

If you serve a file download over PHP with fread and print/echo and experience corrupted binary files, chances are the server still uses magic quotes and escapes the null bytes in your file. Although from 5.3.0 magic quotes are no longer supported, you might still encounter this problem. Try to turn them off by placing this code before using fread:

<?php
@ini\_set('magic\_quotes\_runtime', 0);
?>

[up](http://php.net/manual/vote-note.php?id=97728&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=97728&page=function.fread&vote=down)

2

[***david at identd dot dyndns dot org*** ¶](http://php.net/manual/en/function.fread.php#97728)

**5 years ago**

Note to IIS admins:

When using PHP via the FastCGI ISAPI extension, there is a script timeout of approximately 1hr that cannot be adjusted. When using PHP via CGI, there is a script timeout that is based upon the value of the CGITimeout configuration option. This value must be set extremely high if you plan to serve large files. An explanation of how to configure this option can be found here: <http://www.iisadmin.co.uk/?p=7> If you do not modify this setting you can expect the above scripts to fail silently once it has hit the default value (30 minutes in my case).

[up](http://php.net/manual/vote-note.php?id=85801&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=85801&page=function.fread&vote=down)

2

[***Anonymous*** ¶](http://php.net/manual/en/function.fread.php#85801)

**7 years ago**

It might be worth noting that if your site uses a front controller with sessions and you send a large file to a user; you should end the session just before sending the file, otherwise the user will not be able to continue continue browsing the site while the file is downloading.

[up](http://php.net/manual/vote-note.php?id=78406&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=78406&page=function.fread&vote=down)

2

[***matt at matt-darby dot com*** ¶](http://php.net/manual/en/function.fread.php#78406)

**8 years ago**

I thought I had an issue where fread() would fail on files > 30M in size. I tried a file\_get\_contents() method with the same results. The issue was not reading the file, but echoing its data back to the browser.

Basically, you need to split up the filedata into manageable chunks before firing it off to the browser:

<?php

$total     = filesize($filepath);
$blocksize = (2 << 20); //2M chunks
$sent      = 0;
$handle    = fopen($filepath, "r");

// Push headers that tell what kind of file is coming down the pike
header('Content-type: '.$content\_type);
header('Content-Disposition: attachment; filename='.$filename);
header('Content-length: '.$filesize \* 1024);

// Now we need to loop through the file and echo out chunks of file data
// Dumping the whole file fails at > 30M!
while($sent < $total){
    echo fread($handle, $blocksize);
    $sent += $blocksize;
}

exit(0);

?>

Hope this helps someone!

[up](http://php.net/manual/vote-note.php?id=115340&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=115340&page=function.fread&vote=down)

1

[***john dot wellesz at teaser dot com*** ¶](http://php.net/manual/en/function.fread.php#115340)

**1 year ago**

Note that fread() will return '' (empty string) when a timeout occurs unlike socket\_read() which returns false...

[up](http://php.net/manual/vote-note.php?id=114350&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=114350&page=function.fread&vote=down)

1

[***fread at imperium dot be*** ¶](http://php.net/manual/en/function.fread.php#114350)

**1 year ago**

My script was based on example 3b, but used up 100% CPU when a timeout occurred that wasn't "seen". This is very bad. So here's my code, hoping this will help people out there with the same problem. Obviously first use $rPage = fsockopen(...) and fwrite($rPage,...) and such, after which:

    $sPage    = ''; // the page goes in here
    $iTimeout    = 5; // set the timeout in seconds
    stream\_set\_timeout($rPage,$iTimeout);
    stream\_set\_blocking($rPage,0);
    $fTimeout    = microtime(true);
    do {
        if (($sRead = fread($rPage,8192))!==false and strlen($sRead)) {
            $sPage    .= $sRead; }
        else {
            usleep(10000); } // 0.01 second
        $aInfo    = stream\_get\_meta\_data($rPage); }
    while (!feof($rPage) and !$aInfo['timed\_out'] and microtime(true)-$fTimeout<$iTimeout);
    fclose($rPage);
    // now simply decompress and unchunk $sPage, if need be

Above code will make sure the timeout is used, because this isn't always detected properly. In addition, the usleep() will keep the CPU in check.

[up](http://php.net/manual/vote-note.php?id=113858&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=113858&page=function.fread&vote=down)

0

[***webmaster at lucabanfi dot it*** ¶](http://php.net/manual/en/function.fread.php#113858)

**2 years ago**

//download data File
$handle = fopen("$url\_here", "rb");
$contents = '';
while (!feof($handle)) {
     $contents .=fread($handle, 8192);
}

fclose($handle);

//write file on disk
$fp = fopen("$filename\_here.ext", "wb");
fwrite($fp, $contents);
fclose($fp);

[up](http://php.net/manual/vote-note.php?id=106999&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=106999&page=function.fread&vote=down)

0

[***andrej dot frelih at gmail dot com*** ¶](http://php.net/manual/en/function.fread.php#106999)

**3 years ago**

Another sample function that supports from/to range requests:

<?php
function download\_file($file\_name) {

    if (!file\_exists($file\_name)) { die("<b>404 File not found!</b>"); }

    $file\_extension = strtolower(substr(strrchr($file\_name,"."),1));
    $file\_size = filesize($file\_name);
    $md5\_sum = md5\_file($file\_name);

   //This will set the Content-Type to the appropriate setting for the file
    switch($file\_extension) {
        case "exe": $ctype="application/octet-stream"; break;
        case "zip": $ctype="application/zip"; break;
        case "mp3": $ctype="audio/mpeg"; break;
        case "mpg":$ctype="video/mpeg"; break;
        case "avi": $ctype="video/x-msvideo"; break;

        //The following are for extensions that shouldn't be downloaded (sensitive stuff, like php files)
        case "php":
        case "htm":
        case "html":
        case "txt": die("<b>Cannot be used for ". $file\_extension ." files!</b>"); break;

        default: $ctype="application/force-download";
    }

    if (isset($\_SERVER['HTTP\_RANGE'])) {
        $partial\_content = true;
        $range = explode("-", $\_SERVER['HTTP\_RANGE']);
        $offset = intval($range[0]);
        $length = intval($range[1]) - $offset;
    }
    else {
        $partial\_content = false;
        $offset = 0;
        $length = $file\_size;
    }

    //read the data from the file
    $handle = fopen($file\_name, 'r');
    $buffer = '';
    fseek($handle, $offset);
    $buffer = fread($handle, $length);
    $md5\_sum = md5($buffer);
    if ($partial\_content) $data\_size = intval($range[1]) - intval($range[0]);
    else $data\_size = $file\_size;
    fclose($handle);

    // send the headers and data
    header("Content-Length: " . $data\_size);
    header("Content-md5: " . $md5\_sum);
    header("Accept-Ranges: bytes");
    if ($partial\_content) header('Content-Range: bytes ' . $offset . '-' . ($offset + $length) . '/' . $file\_size);
    header("Connection: close");
    header("Content-type: " . $ctype);
    header('Content-Disposition: attachment; filename=' . $file\_name);
    echo $buffer;
    flush();
}
?>

[up](http://php.net/manual/vote-note.php?id=101677&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=101677&page=function.fread&vote=down)

0

[***ricardsss at gmail dot com*** ¶](http://php.net/manual/en/function.fread.php#101677)

**4 years ago**

Simple userlist script that reads content from a file and prints it back.

<?php

    $file = "users.txt";
    $handle = fopen($file, "a+");

    if(!filesize($file)>0) {
        echo "File is empty!";
    }
    else {
        $fcontent = fread($handle, filesize($file));
        echoUsers();
    }

    fclose($handle);

    function echoUsers() {
        global $fcontent;
        $users = explode(" ", $fcontent);

        foreach($users as $user) {
            echo $user."<br />";
        }
    }

?>

[up](http://php.net/manual/vote-note.php?id=100040&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=100040&page=function.fread&vote=down)

0

[***NOopensourceSPAM at prodigy7 dot de*** ¶](http://php.net/manual/en/function.fread.php#100040)

**5 years ago**

Somehow all code samples for downloads, described here, doesn't work right for me.
When I download a big file readfile or fread in b mode, the final file hasn't the same md5 like the originial.

Some tests helps me, finding a solution:

<?php
            $fp = fopen($DownloadFile, 'rb');
            while ( $cline = fgets($fp) )
            {
                print $cline;
            }
            fclose($fp);
?>

Somehow, it's "binary safe" and deliver that file which are read. md5 original and download are the same.

[up](http://php.net/manual/vote-note.php?id=88175&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=88175&page=function.fread&vote=down)

0

[***James Ranson*** ¶](http://php.net/manual/en/function.fread.php#88175)

**6 years ago**

Tom, the idea for the examples below is to ensure the user has proper credentials before serving the file. With that security in mind, the suggestion of a 302 redirection seems like a risky idea. Anyone with a modicum of networking experience can run a TCP trace and see the 302 Redirect response, as it is actually a response received by the client browser; the browser then makes a subsequent http request for the URL provided in the Location header.  When that 302 response is captured by wireshark, the 'secret' location is then exposed and can be shared with anyone who wishes to bypass the authorization routines in the php.

The only way to secure this would be for the 302 Redirection response to include some kind of unique, per-request, expiring authorization token, either on the end of the url or in a set-cookie, that is then checked by an authorization module implemented within the hosting webserver. Otherwise, you're relegated to the methods described below.

[up](http://php.net/manual/vote-note.php?id=86680&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=86680&page=function.fread&vote=down)

0

[***tom*** ¶](http://php.net/manual/en/function.fread.php#86680)

**7 years ago**

Various scripts suggested here attempt to deliver a file for download to a client. Handling http protocol features such as HTTP\_RANGE is not trivial; neither is handling flow control with the server, memory and time limits when the files are large.

An alternative is to let the web server can handle http by redirecting to the file in question. It's not uncommon e.g. <http://www.apple.com/downloads/macosx/> does this.

A PHP script can do any checks needed (security, authentication, validate the file) and any other tasks before calling header("Location $urltofile");

I tested this with apache. Interrupt/resume download works. The server's mime type configuration will determine client behavior. For apache, if defaults in mime.types are not suitable, configuration directives for mod\_mime could go in a .htaccess file in the directory of the file to download. If really necessary, these could even by written by the PHP script before it redirects.

[up](http://php.net/manual/vote-note.php?id=86633&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=86633&page=function.fread&vote=down)

0

[***Anonymous*** ¶](http://php.net/manual/en/function.fread.php#86633)

**7 years ago**

This code is buggy
<?php
$contents = '';
while (!feof($handle)) {
  $contents .= fread($handle, 8192);
}
?>
When you read a file whose size is a multiple of the readsize (8192 here), then the loop is executed when there are no more data to read. Here, the result of fread() is not checked, and so the instruction
<?php
$contents .= fread($handle, 8192)
?>
is executed once with no data from fread(). In this very case, it is not important, but in some situation it could be harmful.

The good way to read a file block by block is :
<?php
while ( ($buf=fread( $handle, 8192 )) != '' ) {
    // Here, $buf is guaranted to contain data
    $contents .= $buf;
}
if($buf===FALSE) {
    echo "THERE WAS AN ERROR READING\n";
}
?>

[up](http://php.net/manual/vote-note.php?id=74162&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=74162&page=function.fread&vote=down)

0

[***aubfre at hotmail dot com*** ¶](http://php.net/manual/en/function.fread.php#74162)

**8 years ago**

Changing the value of $length may yield to different download speeds when serving a file from a script.

I was not able to max out my 10mbps connection when 4096 was used. I found out that using 16384 would use all the available bandwidth.

When outputing binary data with fread, do not assume that 4096 or 8192 is the optimal value for you. Do some benchmarks by downloading files through your script.

[up](http://php.net/manual/vote-note.php?id=54350&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=54350&page=function.fread&vote=down)

0

[***Richard Dale richard at premiumdata dot n dot e dot t*** ¶](http://php.net/manual/en/function.fread.php#54350)

**10 years ago**

If you use any of the above code for downloadinng files, Internet Explorer will change the filename if it has multiple periods in it to something with square brackets.  To work around this, we check to see if the User Agent contains MSIE and rewrite the necessary periods as %2E

<?php
# eg. $filename="setup.abc.exe";
if (strstr($\_SERVER['HTTP\_USER\_AGENT'], "MSIE")) {
    # workaround for IE filename bug with multiple periods / multiple dots in filename
    # that adds square brackets to filename - eg. setup.abc.exe becomes setup[1].abc.exe
    $iefilename = preg\_replace('/\./', '%2e', $filename, substr\_count($filename, '.') - 1);
    header("Content-Disposition: attachment; filename=$iefilename" );
} else {
    header("Content-Disposition: attachment; filename=$filename");
}
?>

[up](http://php.net/manual/vote-note.php?id=52038&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=52038&page=function.fread&vote=down)

0

[***james at reflexive dot net*** ¶](http://php.net/manual/en/function.fread.php#52038)

**10 years ago**

Several of these examples use a Content-Disposition header to force the browser to save a file but then they specify the file name without quotes. This will cause problems for some browsers (Mozilla Fire Fox) if the file name contains a space.  You must put quotes around the name if you want to work reliably for all files in all browsers.
<?php
header ("Content-Disposition: attachment; filename=$theFileName"); // bad

header ("Content-Disposition: attachment; filename=\"$theFileName\""); // good

?>

[up](http://php.net/manual/vote-note.php?id=51720&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=51720&page=function.fread&vote=down)

0

[***planetiss at gmail dot com*** ¶](http://php.net/manual/en/function.fread.php#51720)

**10 years ago**

For download the big files (more than 8MB), you must used ob\_flush() because the function flush empty the Apache memory and not PHP memory.
And the max size of PHP memory is 8MB, but ob\_flush is able to empty the PHP memory.

<?php
header('Content-Type: application/force-download');
header ("Content-Length: " . filesize($file));
header ("Content-Disposition: attachment; filename=$theFileName");

   $fd = fopen($file, "r");
   while(!feof($fd))
  {
       echo fread($fd, 4096);
       ob\_flush();

   }
?>

[up](http://php.net/manual/vote-note.php?id=111258&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=111258&page=function.fread&vote=down)

-1

[***box dot afzar at gmail dot com*** ¶](http://php.net/manual/en/function.fread.php#111258)

**2 years ago**

I write this script for download with resume suport
<?php
// If user click the download link
if(isset($\_GET['filename'])){
        // The directory of downloadable files
        // This directory should be unaccessible from web
        $file\_dir="/tmp/";

        // Replace the slash and backslash character with empty string
        // The slash and backslash character can be dangerous
        $file\_name=str\_replace("/", "", $\_GET['filename']);
        $file\_name=str\_replace("\\", "", $file\_name);

        // If the requested file is exist
        if(file\_exists($file\_dir.$file\_name)){
                // Get the file size
                $file\_size=filesize($file\_dir.$file\_name);
                // Open the file
                $fh=fopen($file\_dir.$file\_name, "r");

                // Download speed in KB/s
                $speed=5;

                // Initialize the range of bytes to be transferred
                $start=0;
                $end=$file\_size-1;

                // Check HTTP\_RANGE variable
                if(isset($\_SERVER['HTTP\_RANGE']) &&
                        preg\_match('/^bytes=(\d+)-(\d\*)/', $\_SERVER['HTTP\_RANGE'], $arr)){

                        // Starting byte
                        $start=$arr[1];
                        if($arr[2]){
                                // Ending byte
                                $end=$arr[2];
                        }
                }

                // Check if starting and ending byte is valid
                if($start>$end || $start>=$file\_size){
                        header("HTTP/1.1 416 Requested Range Not Satisfiable");
                        header("Content-Length: 0");
                }
                else{
                        // For the first time download
                        if($start==0 && $end==$file\_size){
                                // Send HTTP OK header
                                header("HTTP/1.1 200 OK");
                        }
                        else{
                                // For resume download
                                // Send Partial Content header
                                header("HTTP/1.1 206 Partial Content");
                                // Send Content-Range header
                                header("Content-Range: bytes ".$start."-".$end."/".$file\_size);
                        }

                        // Bytes left
                        $left=$end-$start+1;

}
?>
<html>
<head>
        <title>Home</title>
</head>
<body>
        <a href="index.php?filename=file.pdf">Download</a>
</body>
</html>

[up](http://php.net/manual/vote-note.php?id=41514&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=41514&page=function.fread&vote=down)

0

[***squeegee*** ¶](http://php.net/manual/en/function.fread.php#41514)

**11 years ago**

fread also works for fsockopen's that are open-ended (no feof) if you know how the last packet for a particular set of data should end. For example, if you sent a command to an nntp server, the reply from the server would end with a dot and a carriage return/linefeed. The connection still stays open for more commands, but doing it this way is more efficient than doing line-by-line fgets until you get to the end of the reply.

<?php
if(($res=nntp\_cmd($conn,"BODY $msgid",222))===false){
    continue;
}else{
    $contents='';
    while(1){
        $packet=fread($conn,8192);
        $contents.=$packet;
        if(substr($packet,-3)==".\r\n")break;
    }
    // do something with $contents
}
?>

[up](http://php.net/manual/vote-note.php?id=32112&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=32112&page=function.fread&vote=down)

0

[***Brian*** ¶](http://php.net/manual/en/function.fread.php#32112)

**12 years ago**

Two quick notes on download prompting...
First, the following line:

<?php
header("Cache-Control: no-cache, must-revalidate");
?>

causes IE6 to prompt you to download the script instead of the output and will fail to connect.  Take out that header and everything works perfectly.

Pragma: no-cache doesn't cause a problem.

Second, Mozilla tries to add .php to the download file name if content-type is application. Changing the content type to the more specific MIME type (such as audio/mpeg) fixes that but causes IE to try its plugins (such as Quicktime).

The fix I found for that to specify attachment instead of inline.  Here's my code: a prompted, small buffer MP3 download:

<?php
function downloadMP3 ($fileDir, $fileName) {
   $completeFilePath=$fileDir.'/'.$fileName;
   header('Pragma: no-cache');
   header("Content-type: audio/mpeg\nContent-Disposition: attachment; filename=\"" . $fileName . "\"\nContent-length: ".(string)(filesize($completeFilePath)));
   $fd=fopen($completeFilePath,'rb');
      while(!feof($fd)) {
         print(fread($fd, 4096));
         flush();
      }
}
?>

[up](http://php.net/manual/vote-note.php?id=26824&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=26824&page=function.fread&vote=down)

0

[***rob at lbox.org*** ¶](http://php.net/manual/en/function.fread.php#26824)

**13 years ago**

I spent a while trying to get this to work so I thought I'd share.

Here's how to read a remote binary file using fread.

<?php
$fp = fopen("<http://www.example.com/img.jpg>", "rb");

if($fp){
while(!feof($fp)) {
     $img = $img . fread($fp, 1024);
}
}
?>

This will read the contents of the file into the var $img 1024 bytes at a time.  I used that number because it seemed safe, but you can increment it all you want I guess.

I don't know if everyone but me gets this, but I thought I'd share since I didn't see anything like it out there.

[up](http://php.net/manual/vote-note.php?id=8085&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=8085&page=function.fread&vote=down)

0

[***heikki dot korpela at wapit dot com*** ¶](http://php.net/manual/en/function.fread.php#8085)

**15 years ago**

Fread is binary-safe IF AND ONLY IF you don't use magic-quotes. If you do, all null bytes will become \0, and you might get surprising results when unpacking.

That is, you would do something like

<?php
set\_magic\_quotes\_runtime(0);
?>

before fread()

and something like

<?php
set\_magic\_quotes\_runtime(get\_magic\_quotes\_gpc()) after.
?>

And, after fread, an unpack would be needed, of course. Surprisingly, pack(), however, does not work quite like in Perl (or perhaps I'm just missing something here) - you can't pack an array directly, but instead you'll have to pack each element seperately to the string:

<?php
foreach ($data as $dec) {
  $data\_output .= pack("C\*", $dec);
}
?>

[up](http://php.net/manual/vote-note.php?id=51703&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=51703&page=function.fread&vote=down)

-1

[***mrhappy[at]dotgeek.org*** ¶](http://php.net/manual/en/function.fread.php#51703)

**10 years ago**

Just a note for anybody trying to implement a php handled download script -

We spent a long time trying to figure out why our code was eating system resources on large files.. Eventually we managed to trace it to output buffering that was being started on every page via an include.. (It was attempting to buffer the entire 600 Megs or whatever size \*before\* sending data to the client) if you have this problem you may want to check that first and either not start buffering or close that in the usual way :)

Hope that prevents somebody spending hours trying to fix an obscure issue.

Regards :)

[up](http://php.net/manual/vote-note.php?id=77548&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=77548&page=function.fread&vote=down)

-1

[***Blagovest Buyukliev*** ¶](http://php.net/manual/en/function.fread.php#77548)

**8 years ago**

Having tried to reliably transfer large amounts of binary data over a latent network, I found out that fread()/fwrite() should never be trusted to read/write the whole block with the exact length specified, even in blocking mode, even for small block lengths.

I came up with these two functions, fully-replaceable and reliable alternatives of fread()/fwrite() in a socket context:

<?php

function fullwrite ($sd, $buf) {
  $total = 0;
  $len = strlen($buf);

  while ($total < $len && ($written = fwrite($sd, $buf))) {
    $total += $written;
    $buf = substr($buf, $written);
  }

  return $total;
}

function fullread ($sd, $len) {
  $ret = '';
  $read = 0;

  while ($read < $len && ($buf = fread($sd, $len - $read))) {
    $read += strlen($buf);
    $ret .= $buf;
  }

  return $ret;
}

?>

The functions are "greedy", i.e. trying to read/write as much data as possible at once. If the call to fread()/fwrite() reads/writes less than expected, then the next iteration eats up the remainder. Very smart as only the largest possible chunks are read/written.

Only in case of a broken pipe fullread()/fullwrite() return less than the specified length. Otherwise it is guaranteed that upon termination

strlen(fullread($sd, $len)) == $len

and

fullwrite($sd, $buf) == strlen($buf)

Works perfectly with a socket descriptor returned from stream\_socket\_client() or fsockopen().

Greetings from Rousse, Bulgaria.

[up](http://php.net/manual/vote-note.php?id=100214&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=100214&page=function.fread&vote=down)

-1

[***Anonymous*** ¶](http://php.net/manual/en/function.fread.php#100214)

**5 years ago**

Stream editor on php

#/bin/php
<?php
if (ftell(STDIN) !== 0) {
    fwrite(STDERR, "Pipe error\n");
    exit(1);
}
$input = '';
while (true) {
    $input = trim(fread(STDIN,10240));
    if (feof(STDIN)) break;
    if ($input === false || strlen($input) === 0) {
        continue;
    }
    // replase '<br />' to 'newline'
    $input = preg\_replace('/<br\s\*\/>/', "\n\t", $input);
    fwrite(STDOUT, $input . "\n");
}
?>

Example:
cat some.html | this\_script.php | grep something

Keyword list: pipe, tail, grep, sed

[up](http://php.net/manual/vote-note.php?id=61371&page=function.fread&vote=up)

[down](http://php.net/manual/vote-note.php?id=61371&page=function.fread&vote=down)

-2

[***yellow1912 at yahoo dot com*** ¶](http://php.net/manual/en/function.fread.php#61371)

**9 years ago**

I tried to use the download resume script below, but it put extreme load on the server for just 1 download only (the file is around 200MB).

Be carefull when you test the script on your server. I'll fgets, or other functions and see if it works.

