

MANIPULAÇÃO DE ARQUIVOS EM PHP

DIOGO CEZAR TEIXEIRA BATISTA
<http://inf.cp.utfpr.edu.br/diogo>
diogo@diogocezar.com

MANIPULAÇÃO DE ARQUIVOS

- PHP define diversas funções para leitura e escrita de arquivos de texto;
- arquivos de texto são ideais para o armazenamento de dados simples;
 - mais complexidade: banco de dados;
- deve-se sempre prestar atenção ao esquema de permissões de arquivos no servidor.

FILE_GET_CONTENTS

- a função `file_get_contents` lê todo o conteúdo de um arquivo de texto, e o armazena em uma string;

```
file_get_contents($arquivo)
```

Retorna o conteúdo do arquivo, ou false, se houver erro.

EXEMPLO FILE_GET_CONTENTS

modelo.html

```
Nome: {nome}<br>
E-mail: {email}<br>
```

pagina.php


```
$texto = file_get_contents('modelo.html');
echo str_replace(
    array('{nome}','{email}'),
    array('Fulano de Tal','fulano@detal.com'),
    $texto
)
```

saída

```
Nome: Fulano de Tal<br>
E-mail: fulano@detal.com<br>
```

FILE_GET_CONTENTS

- *file_get_contents* pode ser usada para obter um arquivo a partir de uma URL externa:



```
$var = file_get_contents('http://www.google.com/');
```

FILE_PUT_CONTENTS

- a função *file_put_contents* salva o conteúdo de uma string em um arquivo de texto local;

```
file_put_contents($arquivo, $conteudo)
```



\$conteudo pode ser um array ou uma string

Retorna o número de bytes escritos no arquivo, ou false, em caso de erro

EXEMPLO FILE_PUT_CONTENTS

```
$linhas = array(  
'Primeira linha.',  
'Segunda linha.',  
'Terceira linha.'  
);  
$bytes = file_put_contents('teste.txt',$linhas);  
if($bytes)  
echo "$bytes bytes escritos";  
else  
echo "Erro!";
```

saída

44 bytes escritos

teste.txt

Primeira linha.Segunda linha.Terceira linha.

EXEMPLO FILE_PUT_CONTENTS

```
$linhas = array(  
'Primeira linha.',  
'Segunda linha.',  
'Terceira linha.'  
);  
$bytes = file_put_contents('teste.txt', implode("\n", $linhas));  
if($bytes)  
echo "$bytes bytes escritos";  
else  
echo "Erro!";
```

saída

46 bytes escritos

teste.txt

Primeira linha.
Segunda linha.
Terceira linha.

LENDO UM ARQUIVO POR PARTES

- é possível ler um arquivo passo a passo, ao invés de carregá-lo todo em memória, de uma só vez;
 - útil para o processamento de arquivos longos;
- funções:
 - *fopen*;
 - *fgets* / *fgetc*;
 - *feof*;
 - *fclose*.

FOPEN

- `fopen($arquivo, $modo);`
- Lê o `$arquivo` especificado;
- Retorna uma *Referência de Arquivo* (`$recurso`), ou `false` em caso de erro;
- `$modo` pode ser:
 - 'r' Somente leitura;
 - 'r+' Leitura e escrita;
 - 'w' Somente escrita. Cria ou sobrescreve o arquivo;
 - 'w+' Leitura e escrita. Cria ou sobrescreve o arquivo;
 - 'a' Somente escrita. Cria ou continua o arquivo;
 - 'a+' Leitura e escrita. Cria ou continua o arquivo.

FGETC

- `fgetc($recurso)`
 - lê e retorna o próximo *caractere* do Recurso de Arquivo especificado.

FGETS

- `fgets($recurso)`
 - lê e retorna a próxima *linha* do Recurso de Arquivo especificado.

FEOF

- `feof($recurso)`
 - Retorna *true*, se o arquivo indicado pelo `$recurso` tiver chegado ao fim e *false*, do contrário.

FCLOSE

- *fclose(\$recurso)*
 - finaliza a manipulação do arquivo e o libera ao sistema operacional.

LENDO POR CARACTERE

modo leitura

```
<?php
$rec = fopen('arquivo.txt', 'r');
$i = 0;
while(!feof($rec)) {
    $linha = fgetc($rec);
    echo "Caractere $i: $linha<br>\n";
    $i++;
}
fclose($rec);
?>
```

Caractere 0: P
Caractere 1: r
Caractere 2: i
Caractere 3: m
Caractere 4: e
Caractere 5: i
...

LENDO POR LINHA

modo leitura

```
<?php
$rec = fopen('arquivo.txt', 'r');
$i = 0;
while(!feof($rec)){
    $linha = fgets($rec);
    echo "Linha $i: $linha<br>\n";
    $i++;
}
fclose($rec);
?>
```

Linha 0: Primeira linha
Linha 1: Segunda linha
Linha 2: Terceira linha

ESCREVENDO UM ARQUIVO POR PARTES

- para escrever em um arquivo passo a passo, usam-se as mesmas funções *fopen* (atenção para as opções!) e *fclose*;
- para escrever no Recurso, usa-se:
 - *fwrite(\$res, \$string)*

ESCREVENDO UM ARQUIVO POR PARTES

modo escrita-leitura

```
$rec = fopen('teste2.txt', 'wr+');  
if(!$rec){  
    echo "Erro!";  
}  
else{  
    fwrite($rec, "The book is on the  
table\n");  
    fwrite($rec, "1 + 1 = " . (1+1));  
    fclose($rec);  
}
```

teste2.txt

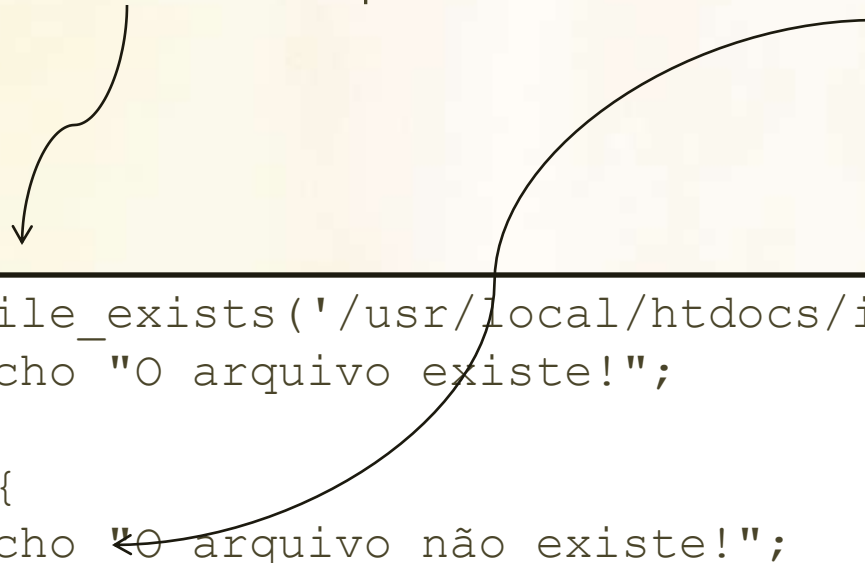
```
The book is on the table  
1 + 1 = 2
```

VERIFICANDO ARQUIVOS

- PHP também possui várias funções voltadas testes realizados sobre arquivos do sistema;
 - existência de arquivos;
 - permissões;
 - tamanho e tipo.

FILE_EXISTS

- *file_exists(\$arquivo)*
 - retorna true se o arquivo existe; false, do contrário.

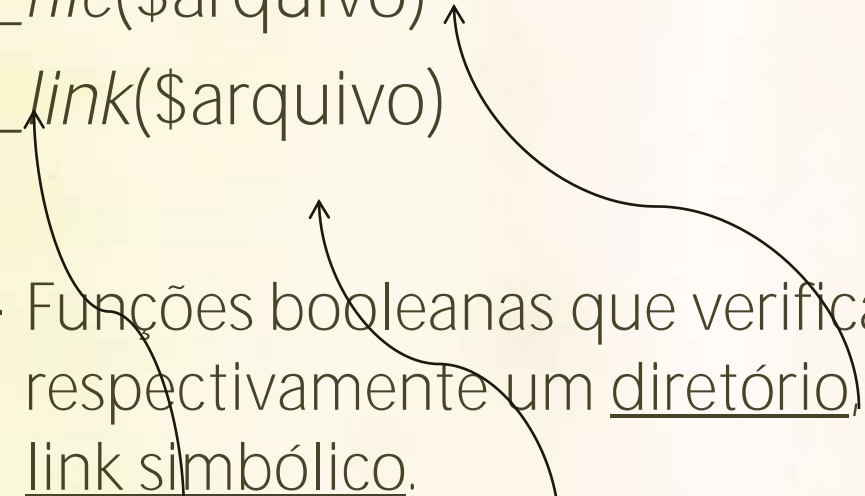


```
if(file_exists('/usr/local/htdocs/index.html')) {
    echo "O arquivo existe!";
}
else{
    echo "O arquivo não existe!";
}
```


PERMISSÕES

- *is_readable(\$arquivo)*
 - *is_writable(\$arquivo)*
 - *is_executable(\$arquivo)*
- funções que retornam *true ou false*, dependendo a permissão verificada (leitura, escrita, execução).

PERMISSÕES

- *is_dir(\$arquivo)*
 - *is_file(\$arquivo)*
 - *is_link(\$arquivo)*
- Funções booleanas que verificam se o arquivo é, respectivamente um diretório, um arquivo comum ou um link simbólico.
- 

DIRETÓRIO E ARQUIVO

- *dirname(\$caminho)*
 - retorna a porção do \$caminho composta pelo diretório;
- *basename(\$caminho, [\$extensao])*
 - retorna a porção do \$caminho composta pelo nome do arquivo;
 - se o nome do arquivo terminar em \$extensao, essa também será eliminada.

TAMANHO DO ARQUIVO

- *filesize(\$arquivo)*
 - retorna o tamanho do arquivo (em bytes).

EXEMPLO

```
<?php Exemplo
function ilista($txt){
echo "<li>O arquivo $txt.</li>\n";
}
$arquivo = '/etc/apt/sources.list';
if(file_exists($arquivo)){
?>
<h3><?= $arquivo ?></h3>
Diretório: <strong><?= dirname($arquivo) ?
></strong><br>
Arquivo: <strong><?= basename($arquivo) ?></
strong><br>
Arquivo sem extensão: <strong><?=
basename($arquivo, '.list') ?></strong><br>
Tamanho: <strong><?= filesize($arquivo) ?></
strong> bytes<br>
```

EXEMPLO

```
<ul>
<?php
if(is_readable($arquivo))  ilista('pode ser
lido');
if(is_writable($arquivo))  ilista('pode ser
escrito');
if(is_executable($arquivo)) ilista('é
executável');
if(is_dir($arquivo))  ilista('é um
diretório');
if(is_file($arquivo))  ilista('é um arquivo
comum');
if(is_link($arquivo))  ilista('é um link
simbólico');
?>
</ul>
```


EXEMPLO

```
<?php
}
else {
echo "Erro: arquivo não existe.<br>";
}
?>
```

**arquivo.txt**

Diretório: .

Arquivo: **arquivo.txt**Arquivo sem extensão: **arquivo.txt**Tamanho: **6** bytes

- O arquivo pode ser lido.
- O arquivo pode ser escrito.
- O arquivo é um arquivo comum.