

# PHP

## Manipulação de Arquivos Texto

Rafael Dias Ribeiro  
rdribeiro@processware.com.br

Sócio Efetivo da:

# Operando sobre arquivos

Abertura:

**recurso** `fopen(string nome_arquivo, string modo [, int usar_include_path [,recurso contexto]])`

**nome\_arquivo**: Nome do arquivo a ser aberto, que pode ser local ou remoto. Se for fornecido um URL, a abertura só irá funcionar se a diretiva `allow_url_fopen` estiver habilitada.

**Modo**: Modo de acesso ao arquivo (lista dos modos no próximo slide)

**usar\_include\_path** : Indica se o arquivo deve ser procurado nos diretórios especificados na diretiva `include_path` no arquivo `php.ini`

**contexto**: Permite a definição de um contexto, que consiste em um conjunto de parâmetros que modificam o comportamento do arquivo. O suporte a contexto foi adicionado no PHP 5.

## Modo

'r' : Abre somente para leitura, posicionando o ponteiro no início do arquivo

'r+' : Abre para leitura e escrita, posicionando o ponteiro no início do arquivo

'w' : Abre somente para escrita, posicionando o ponteiro no início do arquivo e deixando-o com tamanho zero. Se o arquivo não existir, tenta criá-lo

'w+' : Abre para leitura e escrita, posicionando o ponteiro no início do arquivo e deixando-o com tamanho zero. Se o arquivo não existir, tenta criá-lo

'a' : Abre somente para escrita, posicionando o ponteiro no final do arquivo. Se o arquivo não existir, tenta criá-lo

'a+' : Abre para leitura e escrita, posicionando o ponteiro no final do arquivo. Se o arquivo não existir, tenta criá-lo

'x' : Cria e abre um arquivo somente para escrita, posicionando o ponteiro no início do arquivo. Se o arquivo já existir, retorna falso (FALSE) e gera um erro tipo E\_WARNING.  
(Modo usado apenas em arquivos locais)

'x+' : Cria e abre um arquivo para leitura e escrita, posicionando o ponteiro no início do arquivo. Se o arquivo já existir, retorna falso (FALSE) e gera um erro tipo E\_WARNING.  
(Modo usado apenas em arquivos locais)

# Operando sobre arquivos

## Fechamento

### **bool fclose (recurso ponteiro\_arquivo)**

Retorna TRUE se o arquivo for fechado com sucesso e FALSE se houver alguma falha

O parâmetro passado para a função fclose deve conter a variável para a qual foi atribuído o resultado da função fopen, ou seja, o ponteiro para o arquivo que foi aberto.

Ex:

<?

```
$ponteiro = fopen('arquivo.txt','r');
```

```
...
```

```
fclose($ponteiro);
```

```
?>
```

# Operando sobre arquivos

## Leitura

**string fread (recurso ponteiro\_arquivo, int tamanho)**

Lê o número de bytes especificado no parâmetro tamanho. A leitura termina quando o número de bytes é lido ou chega-se ao final do arquivo (EOF – End Of File).

Ex:

```
<?  
$ponteiro = fopen("teste.txt","r");  
$conteudo = fread($ponteiro,30);  
echo $conteudo;  
fclose($ponteiro);  
?>
```

# Operando sobre arquivos

## Leitura

**string fgets (recurso ponteiro\_arquivo [, int tamanho])**

Lê o número de bytes especificado no parâmetro tamanho ou quando terminar a linha atual do arquivo (caracter \n) ou quando chega-se ao final do arquivo (EOF – End Of File).

Ex:

```
<?  
$ponteiro = fopen("teste.txt","r");  
$linha = fgets($ponteiro,4096);  
echo $linha;  
fclose($ponteiro);  
?>
```

# Operando sobre arquivos

## Leitura

### **string fgetc (recurso ponteiro\_arquivo)**

Retorna uma string contendo somente um caracter lido do arquivo apontado por recurso.  
Retorna **FALSE** no fim do arquivo (EOF). .

Ex:

```
<?  
$ponteiro = fopen("teste.txt","r");  
$caracter = fgetc($ponteiro);  
echo $caracter;  
fclose($ponteiro);  
?>
```

# Operando sobre arquivos

## Leitura (Eliminando TAGS HTML)

**string fgetss (recurso ponteiro\_arquivo [, int tamanho])**

Lê a linha do arquivo eliminando as TAGS HTML do arquivo

Ex:

```
<?
$ponteiro = fopen("arquivo1.txt","r");
$linha = fgetss($ponteiro,30);
echo $linha;
fclose($ponteiro);
?>
```



# Operando sobre arquivos

## Leitura e Escrita

`string file_get_contents (string nome_arquivo [, int usar_include_path [, recurso contexto]])`

**nome\_arquivo** : Nome do arquivo a ser aberto, que pode ser local ou remoto. Se for fornecido um URL, a abertura só irá funcionar se a diretiva `allow_url_fopen` estiver habilitada.

**usar\_include\_path** : Indica se o arquivo deve ser procurado nos diretórios especificados na diretiva `include_path` no arquivo `php.ini`

**contexto**: Permite a definição de um contexto, que consiste em um conjunto de parâmetros que modificam o comportamento do arquivo. O suporte a contexto foi adicionado no PHP 5

Esta função oferece um modo mais simples de executar as operações de abertura (`fopen`), leitura (`fread`) e fechamento (`fclose`) de um arquivo, pois essas três operações são realizadas com uma única chamada

# Operando sobre arquivos

## Leitura e Escrita

```
<?  
$conteudo = file_get_contents("arquivo.txt");  
echo $conteudo;  
?>
```

passo 1: - ABRE O ARQUIVO.TXT

passo 2: - LÊ O CONTEÚDO DO ARQUIVO.TXT

passo 3: - ARMAZENA O CONTEÚDO NA VARIÁVEL \$conteudo

passo 4: - FECHA O ARQUIVO.TXT

# Operando sobre arquivos

## Leitura e Escrita – Armazenado o conteúdo em um Array

```
array file (string nome_arquivo [, int usar_include_path [, recurso contexto ]]))
```

//Mesmos parâmetros do file\_get\_contents

Irá armazenar cada linha do arquivo em uma posição do array, assim a posição \$array[0] é igual a primeira linha do arquivo texto, a posição \$array[1] é igual a segunda linha, . . .

Ex:

```
<?
$array = file("arquivo.txt");
foreach ($array as $linha)
echo $linha."<br>";
?>
```

# Operando sobre arquivos

## Exibindo o conteúdo diretamente na tela

```
int readfile (string nome_arquivo [, int usar_include_path [, recurso contexto ]]))
```

```
//Mesmos parâmetros do file_get_contents
```

Ex:

```
<?  
$bytes = readfile("arquivo.txt");  
?>
```

Irá abrir o arquivo, exibir todo seu conteúdo e depois irá fechá-lo.

# Operando sobre arquivos

## Escrita

`int fwrite (recurso ponteiro_arquivo, string string [, int tamanho])`

Ex:

```
<?
$conteudo = "Texto que será gravado dentro do arquivo";
$ponteiro = fopen("arquivo3.txt",w);
fwrite($ponteiro,$conteudo );
fclose($ponteiro);
?>
```

# Operando sobre arquivos

## Excluindo o conteúdo do arquivo

bool ftruncate (recurso ponteiro\_arquivo, int tamanho)

Obs: para excluir todo o seu conteúdo, mantenha o tamanho como zero ( 0 )

ex:

<?

```
$arquivo = "contador.txt";
```

```
$ponteiro = fopen($arquivo,"w");
```

```
ftruncate ($ponteiro,0);
```

```
fclose($ponteiro);
```

?>

# Operando sobre arquivos

## Excluindo o arquivo

`bool unlink (string nome_arquivo)`

ex:

`<?`

`unlink("contador.txt");`

`?>`

Obs: Retorna **True** caso exclua ou **False** caso haja algum problema.  
Esta exclusão funciona para qualquer tipo de arquivo

# ATENÇÃO ESPECIAL PARA ACESSO CONCORRENTE

## flock

Bloqueia ou desbloqueia um arquivo para realizar determinadas operações. É utilizado para evitar escritas concorrentes.

```
bool flock(recurso ponteiro_arquivo, int operação [, int &bloquear])
```

```
ex:  
<?  
$fp = fopen("/tmp/lock.txt", "w+");  
  
if (flock($fp, LOCK_EX)) { // faça um lock exclusivo  
    fwrite($fp, "escrevemos algo aqui\n");  
    flock($fp, LOCK_UN); // libera o lock  
} else {  
    echo "Não pude travar o arquivo !";  
}  
fclose($fp);  
?>
```