Enviar arquivos? Faça Upload!

Provavelmente você não ouve o termo *upload* com tanta frequência como o termo *download*. Ao contrário da operação de *download*, que consiste em "baixar" um arquivo de um computador remoto para seu computador local, fazer *upload* é exatamente o contrário. Consiste em enviar um ou mais arquivos de um computador local para um computador remoto. Neste capítulo, você irá aprender passo a passo como usar os recursos do PHP para fazer o *upload* de arquivos, além de saber em que ocasiões utilizar essa operação.

Por que fazer upload de arquivos?

Muitas vezes, você pode se deparar com a necessidade de descarregar arquivos em um computador remoto. A seguir, serão apresentadas algumas situações em que isso ocorre.

- **Loja virtual**: um exemplo que pode ser citado, e talvez o mais comum, é o cadastramento de produtos em uma loja virtual. Ao inserir um novo produto na loja, é bastante interessante haver a opção de enviar a imagem do produto com seus dados. Assim, qualquer cliente que acessar a loja virtual terá à disposição não só as informações detalhadas de todos os produtos, mas também suas fotos, o que aumentará as chances de efetivação de uma compra.
- **Sites de currículos**: por exemplo, se você tiver um site que permite o cadastramento do currículo das pessoas, um recurso interessante a ser oferecido é o envio das fotos das pessoas

cadastradas. Assim, ao inserir seu currículo, a pessoa teria a opção de "anexar" a seus dados um arquivo contendo sua foto. Esse arquivo ficaria armazenado em algum diretório localizado no servidor e a imagem seria exibida toda vez que alguém consultasse o currículo dessa pessoa.

- **Sites de relacionamentos**: são cada vez mais comuns na internet. O usuário se cadastra preenchendo um formulário com seus dados pessoais, preferências, características da pessoa que procura e, assim como em alguns sites de currículos, é possível enviar uma foto, que poderá ser vista por todas as pessoas que consultarem o perfil desse usuário.
- **Compartilhamento de arquivos**: criar um espaço para os usuários de um site compartilharem arquivos é bastante comum. Pode-se criar, por exemplo, um diretório protegido por senha, para que os usuários de uma lista de discussão possam depositar os arquivos que desejam compartilhar com os demais usuários. O envio desses arquivos pode ser feito com um simples script PHP.
- **Disco virtual**: provavelmente você já deve ter ouvido falar dos discos virtuais oferecidos por diversos provedores de acesso do Brasil, como Terra e UOL. Trata-se de um espaço para você guardar seus arquivos, a fim de manter uma cópia on-line dos mais importantes e acessá-los em qualquer lugar e a qualquer momento. O disco virtual pode relacionar-se ao tópico anterior, visto que também pode ser usado para compartilhar arquivos com outras pessoas. Em ambos os casos, o envio desses arquivos para o servidor é feito por meio de uma operação de upload pela web.
- **Provedores de hospedagem**: alguns provedores de hospedagem oferecem aos webmasters a opção de publicar suas páginas na internet por meio de uma interface web que permite a realização de uploads. Outros oferecem apenas o upload de arquivos com o uso de algum software de FTP (File Transfer Protocol).

Diferenças entre os protocolos HTTP e FTP

Os dois principais protocolos utilizados para realizar transferência de arquivos são o HTTP (*HyperText Transfer Protocol*) e o FTP (*File Transfer Protocol*). O envio por HTTP é feito através de uma página web, enquanto o por FTP ocorre por meio de algum software que possua as funcionalidades de um cliente de FTP.

Como veremos mais adiante, para enviar um arquivo a um programa PHP, precisamos definir uma página web que contenha um formulário HTML com um campo de entrada do tipo `file`, onde o usuário poderá selecionar um arquivo para envio. Isso significa que o arquivo chega até o programa PHP através de uma transferência feita sobre o protocolo HTTP. Quando o arquivo é recebido pelo programa PHP localizado no servidor, podemos definir seu destino de duas formas:

1. simplesmente o movendo para o diretório destino. Nesse caso, teríamos uma simples transferência por HTTP;
2. usando as funções do PHP relativas ao protocolo FTP para conectar-se a um servidor FTP (que pode estar na mesma máquina) e transferir esse arquivo.

O PHP nos disponibiliza uma série de funções para abrir uma conexão FTP de dentro de uma página web, requisitada via HTTP. Ou seja, dentro de nosso programa em PHP, temos à disposição todas as funcionalidades básicas de um software cliente de FTP.

Mas, afinal, qual é a diferença de utilização entre esses dois protocolos? Quando se deve utilizar um ou outro? Na verdade, você não terá que optar por um ou outro protocolo. Como o envio irá partir de uma página web, o protocolo HTTP será sempre usado para transferir o arquivo do computador do usuário para o servidor. Resta a você decidir se para mover o arquivo a seu local destino será usado ou não o protocolo FTP.

Utilizamos comandos FTP quando o arquivo deve ser movido para uma área de FTP no próprio servidor que o recebeu ou, então, para transferir esse arquivo para uma área de FTP localizada em outro servidor. Por exemplo, se no sistema operacional existir um diretório cujo proprietário (*owner*) é um determinado usuário FTP e esse diretório não tem permissão total de escrita, não adiantará tentar simplesmente mover o arquivo recebido para esse local, pois o PHP não terá permissão para isso. A única forma de gravar um arquivo nesse local é fazendo uma transferência via FTP, onde serão fornecidos os dados para autenticação do usuário proprietário do diretório. Como veremos mais adiante, a não ser que seja um FTP anônimo, sempre deverão ser fornecidos um nome de usuário e senha como parâmetros de conexão.

Neste capítulo, serão mostradas as duas formas de definir o destino de um arquivo recebido. Você irá aprender a enviar um arquivo pelo próprio programa navegador, como se fosse um simples campo de um formulário HTML, e logo após movê-lo para o diretório adequado. Da mesma forma, será feito posteriormente no uso do protocolo FTP, no qual iremos alterar os comandos utilizados para definir o destino do arquivo.

Upload utilizando PHP e HTTP

Para fazer um upload utilizando somente o protocolo HTTP, pode-se criar uma única página PHP responsável por toda a operação. Porém, para facilitar o entendimento, vamos definir três arquivos separados para realizar o processo completo:

Arquivo	Descrição
`upload.html`	Página que você deve abrir no navegador. Contém o formulário HTML a partir do qual será selecionado o arquivo a ser enviado ao servidor.

`executa_upload.php`	Página PHP responsável pela operação de upload. É ativada pelo formulário da página `upload.html`, recebendo o arquivo especificado pelo usuário.
`config_upload.inc`	Arquivo contendo os parâmetros de configuração a ser considerados pela página `executa_upload.php`.

Definindo o formulário HTML

O primeiro passo para permitir ao usuário fazer um upload é a criação de um formulário HTML, pelo qual ele irá selecionar o arquivo a ser enviado. A linguagem HTML já nos disponibiliza, entre seus tipos de campos, o tipo `file` para o envio de arquivos. Veja, a seguir, o código-fonte da página `upload.html`, que contém o formulário de envio:

upload.html

```
<html>
<head>
<title>Livro de PHP - Upload de arquivo</title>
</head>
<body>
<h2 align="center">Upload de Arquivos</h2>
<form method="POST" action="executa_upload.php" enctype="multipart/form-data">
<input type="hidden" name="MAX_FILE_SIZE" value="2000000">
<p align=center>Arquivo: <input type="file" name="arquivo" size="30">
<p align=center><input type=submit value="Enviar arquivo">
</form>
</body>
</html>
```

Ao declararmos no formulário um campo de entrada (*input*) do tipo `file`, automaticamente serão

exibidos, no navegador, uma caixa de texto e um botão que, quando pressionado, abrirá uma janela para a seleção de um arquivo qualquer no computador do usuário. A figura 6.1 mostra essa página sendo exibida no programa navegador.

Figura 6.1 – Selecionando um arquivo para upload.

Agora veremos uma rápida explicação sobre as três principais linhas do arquivo `upload.html`. A primeira é a linha onde o formulário é definido:

```
<form method="POST" action="executa_upload.php" enctype="multipart/form-data">
```

Na opção `action` da tag `form`, especifica-se o nome do programa PHP que será ativado pelo formulário quando o usuário clicar o botão de envio. Para indicar que irá ocorrer o upload de um arquivo para o servidor, deve-se utilizar a opção `enctype` com o valor `multipart/form-data`.

Observe agora a linha seguinte:

```
<input type="hidden" name="MAX_FILE_SIZE" value="2000000">
```

Em operações de upload, o parâmetro `MAX_FILE_SIZE` é utilizado pelo programa navegador para definir um tamanho máximo (em bytes) para o arquivo a ser enviado.

Dessa forma, pode-se prevenir que o usuário selecione um arquivo muito grande e consuma muitos recursos do servidor. Essa é uma validação feita no lado cliente, ou seja, o envio do arquivo ao servidor só ocorrerá se seu tamanho for inferior ou igual ao valor do parâmetro `MAX_FILE_SIZE` (em nosso exemplo, inferior ou igual a 2.000.000 bytes, ou seja, aproximadamente 2 megabytes).

Como veremos mais adiante, essa validação também pode ser feita no lado do servidor. Porém, para validar o tamanho do arquivo no servidor, ele deve primeiro chegar até lá. Por exemplo, se o tamanho limite for 2.000.000 bytes e o usuário escolher um arquivo de 3.000.000 bytes, esse arquivo será primeiramente transferido para o servidor, para somente depois o programa detectar que seu tamanho é inválido.

De qualquer forma, é bom que sejam feitas as duas validações, isto é, tanto no lado do cliente como no lado do servidor. Assim, você estará protegido caso algum usuário tente burlar o formulário HTML aumentando o valor ou removendo o parâmetro de limitação de tamanho. Por fim, observe a próxima linha:

```
<p align=center>Arquivo: <input type="file" name="arquivo" size="30">
```

Nessa linha é definido o campo de entrada do tipo `file`, que irá possibilitar ao usuário a seleção de um arquivo qualquer.

Parâmetros de configuração do upload

Antes de começar o upload de seus arquivos, você pode configurar suas preferências em um arquivo que nomeamos como `config_upload.inc`. Trata-se basicamente de um arquivo PHP contendo algumas variáveis que serão utilizadas pelo programa que executa o upload. Utilizou-se a extensão `.inc` apenas para indicar que esse arquivo será incluído em outra página, porém você pode alterá-la caso deseje. Acompanhe a seguir o código-fonte e logo após uma explicação sobre cada uma das variáveis:

config_upload.inc

```
<?php
// parâmetros de configuração da operação
// limitar as extensões? (sim ou nao)
$limitar_ext = "sim";
// extensões autorizadas
$extensoes_validas = array(".gif",".jpg",".jpeg",".bmp");
// caminho absoluto onde os arquivos serão armazenados
$caminho_absoluto = "c:\apache\upload";
// limitar o tamanho do arquivo? (sim ou nao)
$limitar_tamanho = "nao";
// tamanho limite do arquivo em bytes
$tamanho_bytes = "2000000";
// se já existir o arquivo, indica se ele deve ser sobrescrito (sim ou nao)
$sobrescrever = "nao";
?>
```

A maioria dos parâmetros trata basicamente das restrições a ser impostas para a execução de uma operação de upload. A seguir será apresentada uma descrição de cada um deles.

Parâmetro	Descrição
`$limitar_ext`	Pode conter os valores "sim" ou "nao". Indica se deve ou não ser consultado o array com as extensões válidas para verificação do tipo de arquivo que está sendo enviado.
`$extensoes_validas`	Array contendo as extensões de arquivo que serão aceitas pelo servidor. Esse array é consultado somente quando o valor da variável `$limitar_ext` for igual a "sim". Nesse caso, se a extensão do arquivo que está sendo enviado não estiver contida no array, o upload será abortado e uma mensagem de erro será exibida.

`$caminho_absoluto`	Define o diretório onde serão gravados os arquivos enviados pelo usuário. Altere o valor dessa variável colocando o caminho completo do diretório destino e certifique-se de que esse diretório tem permissão de escrita. Por exemplo, no Linux, deve-se dar permissão de escrita ao diretório digitando `chmod 777 nome_diretorio`, através da linha de comando. Em nosso exemplo, utilizou-se o valor "`c:\apache\upload`", pois o programa foi testado nesse diretório em ambiente Windows. Se você quiser que o arquivo seja gravado na própria pasta onde o programa será executado, coloque o valor "." nesta variável, pois o ponto representa o diretório corrente.
`$limitar_tamanho`	Pode conter os valores "sim" ou "nao". Indica se deve ou não ser consultada a variável `$tamanho_bytes` para limitar o tamanho do arquivo que está sendo enviado.
`$tamanho_bytes`	Consultada apenas quando o valor da variável `$limitar_tamanho` for igual a "sim". Nesse caso, se o tamanho do arquivo que está sendo enviado for maior do que o valor definido nessa variável, o upload será abortado e uma mensagem de erro será exibida.
`$sobrescrever`	Pode conter os valores "sim" ou "nao". Indica o que deve ser feito quando já existir um arquivo destino com o mesmo nome do que está sendo enviado. Se for definido o valor "sim", o arquivo destino será sobrescrito, caso contrário será exibida uma mensagem informando que já existe um arquivo de mesmo nome.

Recebendo e gravando o arquivo

A seguir será apresentado o programa responsável pela conclusão da operação de upload. Esse programa recebe o arquivo enviado pelo formulário da página `upload.html`, faz as devidas consistências e grava o arquivo no local destino. Acompanhe o código-fonte e logo após leia as explicações sobre os principais trechos do programa:

executa_upload.php

```
<html>
```

```
<head>
<title>Livro de PHP - Upload de arquivo</title>
</head>
<body>
<h2 align="center">Upload de Arquivos</h2>
<?php
set_time_limit (0);
include 'config_upload.inc';
$nome_arquivo = $_FILES['arquivo']['name'];
$tamanho_arquivo = $_FILES['arquivo']['size'];
$arquivo_temporario = $_FILES['arquivo']['tmp_name'];
if (!empty ($nome_arquivo)) {
    if ($sobrescrever == "nao" && file_exists("$caminho_absoluto/$nome_arquivo"))
        die("Arquivo já existe.");
    if (($limitar_tamanho == "sim") && ($tamanho_arquivo > $tamanho_bytes))
        die("Arquivo deve ter no máximo $tamanho_bytes bytes.");
    $ext = strrchr($nome_arquivo,'.');
    if ($limitar_ext == "sim" && !in_array($ext,$extensoes_validas))
        die("Extensão de arquivo inválida para upload.");
    if (move_uploaded_file($arquivo_temporario, "$caminho_absoluto/$nome_arquivo")) {
        echo "<p align=center>O upload do arquivo <b>". $nome_arquivo."
        </b> foi concluído com sucesso.</p>";
        echo "<p align=center><a href='upload.html'>Novo upload</a></p>";
    }
    else
        echo "<p align=center>O arquivo não pode ser copiado para o servidor.</p>";
}
else
    die("Selecione o arquivo a ser enviado");
```

```
?>
</body>
</html>
```

Vamos analisar agora os principais trechos do código. No início, o comando `set_time_limit (0)` foi utilizado para eliminar o limite de tempo de execução do programa. Se esse comando não for utilizado, será considerado o valor-padrão (30 segundos) ou, então, se existir, o valor do parâmetro `max_execution_time` do arquivo de configuração `php.ini`. Se o arquivo a ser transferido for um pouco grande e seu tempo de transferência ultrapassar esse valor, o upload não ocorrerá com sucesso, daí a importância da eliminação do limite.

Logo após foi utilizado o comando `include` para incluir o arquivo que contém as variáveis de configuração da operação. Isso foi feito por meio da seguinte linha:

```
include 'config_upload.inc';
```

Nas três linhas seguintes, obtemos as informações sobre o arquivo que foi enviado pelo usuário. São elas:

```
$nome_arquivo = $_FILES['arquivo']['name'];
$tamanho_arquivo = $_FILES['arquivo']['size'];
$arquivo_temporario = $_FILES['arquivo']['tmp_name'];
```

Nessas três linhas, obtemos primeiro o nome do arquivo exatamente como está na máquina do usuário, depois o tamanho desse arquivo e, por último, o nome temporário que o servidor utilizou para armazenar o arquivo.

Veja que, no exemplo apresentado, se utilizou o array superglobal (disponível em todo o escopo do programa) `$_FILES`. Note que esse array tem duas dimensões. A primeira é o identificador do arquivo, isto é, o nome do campo do tipo `file` definido no formulário HTML. A segunda dimensão é o nome da propriedade cujo valor desejamos obter. No nosso exemplo, teríamos à

disposição os seguintes elementos:

Elemento	Descrição
`$_FILES['arquivo']['name']`	O nome original do arquivo no computador do usuário.
`$_FILES['arquivo']['type']`	Se existir, contém o tipo MIME (Multipurpose Internet Mail Extensions) do arquivo. Por exemplo, para uma imagem GIF, teríamos o valor `image/gif`.
`$_FILES['arquivo']['size']`	O tamanho do arquivo em bytes.
`$_FILES['arquivo']['tmp_name']`	Nome temporário que o servidor utilizou para armazenar o arquivo.
`$_FILES['arquivo']['error']`	Armazena o código do erro em caso de falha na operação de upload do arquivo.

Prosseguindo com a análise do código do programa `executa_upload.php`, após a obtenção das informações sobre o arquivo, iniciam-se as consistências baseadas nas variáveis definidas em nosso arquivo de configuração `config_upload.inc`. A primeira é a seguinte:

```
if ($sobrescrever == "nao" && file_exists("$caminho_absoluto/$nome_arquivo"))
```

Se você definir o valor "nao" para a variável `$sobrescrever`, o script irá prosseguir somente se não existir um arquivo com mesmo nome no local destino, caso contrário será exibida uma mensagem de erro. Logo após temos a consistência relativa ao tamanho do arquivo:

```
if (($limitar_tamanho == "sim") && ($tamanho_arquivo > $tamanho_bytes))
```

Lembre-se da importância de fazer essa consistência de tamanho no lado do servidor, visto que é fácil de enganar o parâmetro `MAX_FILE_SIZE` do formulário HTML do cliente. A próxima validação

é referente à extensão do arquivo que está sendo enviado:

```
$ext = strrchr($nome_arquivo,'.');
if ($limitar_ext == "sim" && !in_array($ext,$extensoes_validas))
```

Utilizamos a função `strrchr` para retornar a porção do nome do arquivo que se inicia no ponto e vai até seu final, ou seja, o ponto mais sua extensão. Logo após, usamos a função `in_array` para verificar se algum elemento do array `$extensoes_validas` contém esse valor. Se não contiver e o parâmetro `$limitar_ext` estiver habilitado (valor "sim"), o upload não será aceito, por não se tratar de um tipo de arquivo permitido.

Caso todas as condições sejam satisfeitas, o arquivo enviado será movido do diretório temporário utilizado pelo servidor para seu destino:

```
move_uploaded_file($arquivo_temporario, "$caminho_absoluto/$nome_arquivo")
```

Se você estiver usando uma versão mais antiga do PHP e o comando `move_uploaded_file` não for reconhecido, você deverá substituí-lo pelo comando `copy`, como mostra a linha a seguir:

```
copy($arquivo_temporario, "$caminho_absoluto/$nome_arquivo");
```

Caso ocorra algum erro na tentativa de mover o arquivo, será exibida a mensagem informando que o arquivo não pôde ser copiado para o servidor. Um exemplo de erro que ocorre com frequência é o de quando o sistema operacional não consegue gravar o arquivo no local destino devido à ausência de permissão de escrita no diretório.

Lembre-se também que este exemplo funciona para arquivos de até 2 megabytes. Caso você queira enviar arquivos maiores, deve aumentar o valor da variável `$tamanho_bytes` no programa de configuração `config_upload.inc`.

Se não ocorrer erro algum, o upload será concluído e será exibida uma mensagem indicando operação bem-sucedida, como mostra a figura 6.2.

Figura 6.2 – Mensagem exibida ao final da operação.

Upload utilizando PHP e FTP

Veremos agora como realizar um upload utilizando as funções que o PHP oferece para uso do protocolo de transferência FTP. Assim como foi feito no upload simples via HTTP, aqui também vamos definir três arquivos que irão realizar o processo completo. São eles:

Arquivo	Descrição
`upload_ftp.html`	É a página que você deve abrir no programa navegador. Contém o formulário HTML, a partir do qual será selecionado o arquivo a ser enviado ao servidor.
`executa_upload_ftp.php`	Página PHP responsável pela operação de upload. É ativada pelo formulário da página `upload_ftp.html`, recebendo o arquivo especificado pelo usuário.

`config_upload_ftp.inc`	Arquivo contendo os parâmetros de configuração do servidor e do upload, que serão utilizados pela página `executa_upload_ftp.php`.

Definindo o formulário HTML

Assim como no upload simples via HTTP, a primeira tarefa a ser realizada é criar um formulário HTML pelo qual o usuário irá selecionar o arquivo a ser enviado. Deve-se utilizar um campo de entrada (*input*) do tipo `file`. O código desse formulário é quase idêntico ao do formulário criado para o upload por HTTP.

Alteraram-se apenas o título da página e o valor da opção `action`, que contém o nome do programa PHP que irá receber o arquivo. O código-fonte é apresentado a seguir:

upload_ftp.html

```
<html>
<head>
<title>Livro de PHP - Upload de arquivo por FTP</title>
</head>
<body>
<h2 align="center">Upload de Arquivo por FTP</h2>
<form method="POST" action="executa_upload_ftp.php" enctype="multipart/form-data">
    <input type="hidden" name="MAX_FILE_SIZE" value="2000000">
    <p align=center>Arquivo: <input type="file" name="arquivo" size="30">
    <p align=center><input type=submit value="Enviar arquivo">
</form>
</body>
</html>
```

Para ver a descrição detalhada sobre o significado das principais linhas, consulte o tópico

correspondente no tópico "Definindo o formulário HTML" do upload simples por HTTP.

Configuração do servidor e do upload

Antes de começar o upload dos arquivos, você pode configurar suas preferências através do arquivo que nomeamos como `config_upload_ftp.inc`. Esse arquivo tem todas as variáveis de configuração que vimos no `config_upload.inc`, além das configurações relativas à conexão com o servidor FTP que receberá o arquivo. Acompanhe a seguir o código-fonte e logo após uma explicação sobre cada uma das variáveis:

config_upload_ftp.inc

```
<?php
// configurações do servidor FTP
$servidor_ftp = "192.168.1.36";
$usario_ftp = "juliano";
$senha_ftp = "12345";
// limitar as extensões? (sim ou nao)
$limitar_ext = "sim";
// extensões autorizadas
$extensoes_validas = array(".gif",".jpg",".jpeg",".bmp");
//caminho absoluto onde os arquivos serão armazenados
$caminho_absoluto = "public_html/upload";
// limitar o tamanho do arquivo? (sim ou nao)
$limitar_tamanho = "nao";
// tamanho limite do arquivo em bytes
$tamanho_bytes = "2000000";
// se já existir o arquivo, indica se ele deve ser sobrescrito (sim ou nao)
$sobrescrever = "nao";
```

```
?>
```

As três primeiras variáveis (`$servidor_ftp`, `$usuario_ftp` e `$senha_ftp`) definem, respectivamente, o endereço do servidor FTP com o qual a conexão será aberta, o nome de usuário e a senha de conexão. Altere estas três variáveis de acordo com os dados de conexão ao seu servidor FTP. Defina também na variável `$caminho_absoluto` o caminho completo do diretório no qual o arquivo será armazenado, a partir do diretório raiz do seu servidor de FTP. Deve ser uma pasta existente e com permissão de escrita.

Para ver a descrição detalhada sobre o significado das demais variáveis, consulte o tópico "Parâmetros de configuração do upload", no upload simples por HTTP.

Conectando ao servidor e enviando o arquivo

A seguir será apresentado o programa responsável pela operação de upload por FTP. Esse programa recebe o arquivo enviado pelo formulário da página `upload_ftp.html`, faz as devidas consistências e envia o arquivo para o local destino. Acompanhe o código-fonte e, logo após, leia as explicações sobre os principais trechos do programa:

executa_upload_ftp.php

```
<html>
<head>
<title>Livro de PHP - Upload de arquivo por FTP</title>
</head>
<body>
<h2 align="center">Upload de Arquivos</h2>
<?php
// elimina o limite de tempo de execução
set_time_limit (0);
```

```
// inclui o arquivo com as configurações
include 'config_upload_ftp.inc';
$nome_arquivo = $_FILES['arquivo']['name'];
$tamanho_arquivo = $_FILES['arquivo']['size'];
$arquivo_temporario = $_FILES['arquivo']['tmp_name'];

if (!empty ($nome_arquivo)) {
    if ($sobrescrever == "nao" && file_exists("$caminho_absoluto/$nome_arquivo"))
        die("Arquivo já existe.");
    if (($limitar_tamanho == "sim") && ($tamanho_arquivo > $tamanho_bytes))
        die("Arquivo deve ter no máximo $tamanho_bytes bytes.");
    $ext = strrchr($nome_arquivo,'.');
    if ($limitar_ext == "sim" && !in_array($ext,$extensoes_validas))
        die("Extensão de arquivo inválida para upload.");

    // abre uma conexão FTP
    $id_conexao = ftp_connect($servidor_ftp);
        // realiza o login com o nome de usuário e senha
        $login = ftp_login($id_conexao, $usario_ftp, $senha_ftp);
        // verifica se houve sucesso na conexão
        if ((!$id_conexao) || (!$login)) {
            echo "Não foi possível abrir conexão FTP com o servidor $servidor_ftp";
            exit();
        }
        else
            echo "<p align='center'>Usuário $usario_ftp conectado ao servidor
  $servidor_ftp</p>";

        // faz o upload do arquivo
```

```
        $arquivo_destino = "$caminho_absoluto/$nome_arquivo";
        $upload = ftp_put($id_conexao, $arquivo_destino, $arquivo_temporario,
    FTP_BINARY);

        // verifica se houve sucesso no upload
        if (!$upload)
            echo "<p align='center'>O upload do arquivo $nome_arquivo falhou!</p>";
        else {
            echo "<p align='center'>O upload do arquivo $nome_arquivo foi concluído com
                sucesso!</p>";
            echo "<p align='center'><a href='upload_ftp.html'>Novo upload</a></p>";
        }

    // Fecha a conexão FTP
    ftp_close($id_conexao);
}
else
    die("Selecione o arquivo a ser enviado");
?>
</body>
</html>
```

Vamos analisar agora os trechos de código novos em relação ao upload feito anteriormente pelo arquivo `executa_upload.php`. O comando `include` foi utilizado para incluir o arquivo que contém as variáveis de configuração do servidor e da operação. Isso foi feito por meio da seguinte linha:

```
include 'config_upload_ftp.inc';
```

A obtenção das informações sobre o arquivo recebido, assim como as consistências em relação às variáveis de configuração, foi feita da mesma forma que o upload simples por HTTP. Logo, para

obter a descrição detalhada sobre esses trechos de código, consulte o tópico "Recebendo e gravando o arquivo".

Após realizadas todas as consistências, iniciamos a conexão FTP através da função `ftp_connect`, que possui a seguinte sintaxe:

```
recurso ftp_connect (string servidor [, int porta [, int tempo]])
```

Parâmetro	Descrição
servidor	Nome ou endereço IP do servidor com o qual a conexão será aberta.
porta	Especifica a porta do sistema operacional na qual será feita a conexão. Se for omitida ou for igual a zero, será usada a porta 21, padrão para conexões FTP.
tempo	Define o tempo máximo de espera para as operações seguintes. Se for omitido, será utilizado o valor-padrão (90 segundos). Esse tempo pode ser consultado ou alterado a qualquer momento com as funções `ftp_set_option` e `ftp_get_option`.

O estabelecimento da conexão FTP foi feito por meio da seguinte linha:

```
$id_conexao = ftp_connect($servidor_ftp);
```

Nesse exemplo, estamos armazenando esse endereço na variável `$servidor_ftp`, que deve ser configurada previamente no arquivo `config_upload_ftp.inc`. O próximo passo é realizar a autenticação no servidor, fornecendo o nome de usuário e senha. Isso é feito com o uso da função `ftp_login`, que possui a seguinte sintaxe:

```
bool ftp_login (recurso id_conexão, string usuário, string senha)
```

Parâmetro	Descrição
id_conexão	Identificador da conexão aberta previamente pelo comando `ftp_connect`.

usuário	Nome de um usuário (username) autorizado no servidor de FTP.
senha	Senha de acesso para o usuário especificado.

Em nosso exemplo, o login foi feito através da linha a seguir. As variáveis `$usuario_ftp` e `$senha_ftp` também devem ser previamente configuradas no arquivo de configuração:

```
$login = ftp_login($id_conexao, $usario_ftp, $senha_ftp);
```

Para verificar se houve sucesso na conexão e a autenticação do usuário, testa-se o conteúdo das variáveis que receberam o resultado das operações:

```
if ((!$id_conexao) || (!$login)) {
    echo "Não foi possível abrir uma conexão FTP com o servidor $servidor_ftp";
    exit;
}
else
    echo "Usuário $usario_ftp conectado ao servidor $servidor_ftp <br>";
```

As funções `ftp_connect` e `ftp_login` retornam o valor `FALSE` em caso de falha. Se houve sucesso, será exibida uma mensagem confirmando o estabelecimento da conexão e, então, utilizamos o comando `ftp_put` para enviar o arquivo por FTP a seu diretório destino. Sua sintaxe é a seguinte:

```
bool ftp_put (recurso id_conexão, string arq_remoto, string arq_local, int modo
    [, int pos_inicial])
```

Parâmetro	Descrição
id_conexão	Identificador da conexão aberta previamente pelo comando `ftp_connect`.
arq_remoto	Define o nome com o qual o arquivo será armazenado no servidor FTP.
arq_local	Nome do arquivo local que será enviado ao servidor FTP.

modo	Modo de transferência. O arquivo pode ser transferido no modo texto (`FTP_ASCII`) ou binário (`FTP_BINARY`).
pos_inicial	Parâmetro opcional. Define a posição inicial do arquivo a partir da qual deve começar o envio.

Para definir o local destino do arquivo que estamos transferindo no programa `executa_upload_ftp.php`, foi feita a concatenação do caminho base (variável `$caminho_absoluto` do arquivo com as variáveis de configuração) com o nome original do arquivo e, logo após, executada a função `ftp_put` para realizar o envio:

```
$arquivo_destino = "$caminho_absoluto/$nome_arquivo";
$upload = ftp_put($id_conexao, $arquivo_destino, $arquivo_temporario, FTP_BINARY);
```

Em caso de falha, a função `ftp_put` retorna `FALSE`. Assim, podemos testar se realmente o upload ocorreu com sucesso, caso contrário exibimos uma mensagem informando sobre a falha na operação:

```
if (!$upload)
    echo "<p align=center>O upload do arquivo $nome_arquivo falhou!</p>";
```

Para finalizar o programa, encerramos a conexão FTP com o comando `ftp_close`, que recebe como parâmetro o identificador da conexão que desejamos finalizar:

```
ftp_close($id_conexao);
```

Enviando múltiplos arquivos

O PHP também possui suporte para o envio de múltiplos arquivos. Se você pretende permitir ao usuário selecionar mais de um arquivo, de forma que o programa fique o mais genérico possível e não haja necessidade de repetição de código, existem duas maneiras de fazer isso. Pode-se utilizar variáveis em sequência ou um array. O uso do array é muito mais simples, porém veremos como

implementar as duas formas.

Utilizando variáveis em sequência

Essa forma de envio consta em definir um nome diferente para cada campo do tipo "`file`" do formulário HTML. Porém, esses nomes devem formar uma sequência definida pelo nome do campo mais um número identificador. Por exemplo, observe a tela mostrada na figura 6.3.

Nomeamos esse arquivo como `upload_multiplo.html`. Acompanhe agora seu código-fonte.

Figura 6.3 – Seleção de múltiplos arquivos para upload.

upload_multiplo.html

```
<html>
<head>
<title>Livro de PHP - Upload de múltiplos arquivos</title>
</head>
<body>
<h2 align="center">Upload de múltiplos arquivos</h2>
<form method=POST action="executa_upload_multiplo.php" enctype=multipart/form-data>
<input type="hidden" name="MAX_FILE_SIZE" value="200000">
<p align=center>Arquivo 1: <input type="file" name="arquivo1" size="30"></p>
<p align=center>Arquivo 2: <input type="file" name="arquivo2" size="30"></p>
<p align=center>Arquivo 3: <input type="file" name="arquivo3" size="30"></p>
<p align=center>Arquivo 4: <input type="file" name="arquivo4" size="30"></p>
<p align=center>Arquivo 5: <input type="file" name="arquivo5" size="30"></p>
<p align=center><input type=submit value="Enviar arquivos"></p>
</form>
</body>
</html>
```

Veja que, no formulário HTML, foram criados cinco campos do tipo `file`, nomeados como `arquivo1`, `arquivo2`, `arquivo3`, `arquivo4` e `arquivo5`. Os nomes dos campos são quase iguais, só alteramos o último dígito, que vamos utilizar como um número identificador dentro do script PHP. A ideia é montar o nome do elemento de forma dinâmica dentro do programa, para que possamos executar cinco vezes o mesmo código, em vez de repeti-lo.

Para configuração da operação, será utilizado o mesmo arquivo `config_upload.inc`, definido no tópico "Parâmetros de configuração do upload". A página que irá receber e processar os arquivos foi nomeada como `executa_upload_multiplo.php` e seu código-fonte será apresentado a seguir. Acompanhe o código e logo após as explicações:

executa_upload_multiplo.php

```
<html>
<head>
<title>Livro de PHP - Upload de múltiplos arquivos</title>
</head>
<body>
<h2 align="center">Upload de múltiplos arquivos</h2>
<?
// elimina o limite de tempo de execução
set_time_limit (0);
// inclui o arquivo com as configurações
include 'config_upload.inc';
// repete os mesmos comandos para os 5 arquivos
for ($i = 1; $i <= 5; $i++) {
    $id_arquivo = "arquivo".$i;
    $erro = FALSE;
    $nome_arquivo = $_FILES[$id_arquivo]['name'];
    $tamanho_arquivo = $_FILES[$id_arquivo]['size'];
    $arquivo_temporario = $_FILES[$id_arquivo]['tmp_name'];
    if (!empty ($nome_arquivo)) {
        if ($sobrescrever == "nao" && file_exists("$caminho_absoluto/$nome_arquivo"))
        {
            $erro = TRUE;
            echo "Arquivo $nome_arquivo já existe.";
        }
        if (($limitar_tamanho == "sim") && ($tamanho_arquivo > $tamanho_bytes)) {
            $erro = TRUE;
            echo "Arquivo $nome_arquivo deve ter no máximo $tamanho_bytes bytes.";
```

```
            }
        $ext = strrchr($nome_arquivo,'.');
        if ($limitar_ext == "sim" && !in_array($ext,$extensoes_validas)) {
            $erro = TRUE;
            echo "Extensão do arquivo $nome_arquivo inválida para upload.";
        }
    if (!$erro && move_uploaded_file($arquivo_temporario,
        "$caminho_absoluto/$nome_arquivo"))
        echo "<p align=center>O upload do arquivo <b>$nome_arquivo
        </b> foi concluído com sucesso.</p>";
    else
        echo "<p align=center>O arquivo $nome_arquivo não pôde ser copiado para o
            servidor.</p>";
    }
}
?>
</body>
</html>
```

Assim como nos outros scripts de upload apresentados neste capítulo, iniciamos eliminando o limite de tempo (`set_time_limit`) e adicionando uma chamada ao arquivo de configuração. Logo após, utilizou-se o comando `for` para criar um laço de repetição, com a variável de controle `$i` iniciando no valor 1 e finalizando no valor 5. Assim podemos usar o mesmo bloco de comandos para manipular todos os arquivos recebidos. Observe agora as linhas seguintes:

```
$id_arquivo = "arquivo".$i;
$erro = FALSE;
$nome_arquivo = $_FILES[$id_arquivo]['name'];
$tamanho_arquivo = $_FILES[$id_arquivo]['size'];
```

```
$arquivo_temporario = $_FILES[$id_arquivo]['tmp_name'];
```

Veja que o identificador do arquivo, que indica o elemento a ser acessado no array `$_FILES`, foi criado dinamicamente. Definimos o prefixo "arquivo", que é concatenado ao valor de `$i` na iteração atual. A variável `$erro` foi definida para indicar quando alguma das consistências falhar, a fim de evitar a cópia do arquivo sem interromper a execução do programa.

Após a realização de todos os testes baseados nas variáveis de configuração, o upload será concluído ou será exibida uma mensagem de erro. Esse procedimento será realizado para cada um dos arquivos recebidos:

```
if (!$erro && move_uploaded_file($arquivo_temporario,
        "$caminho_absoluto/$nome_arquivo"))
    echo "<p align=center>O upload do arquivo <b>$nome_arquivo</b> foi concluído com
        sucesso.</p>";
else
    echo "<p align=center>O arquivo $nome_arquivo não pôde ser copiado para o servidor.
  </p>";
```

Note que o usuário não é obrigado a escolher cinco arquivos para enviar. O programa irá tentar localizar os identificadores `arquivo1`, `arquivo2`, `arquivo3`, `arquivo4` e `arquivo5`. Caso não encontre nenhum deles, simplesmente não serão executados as consistências nem o upload. Isso significa que se o usuário preencher, por exemplo, apenas o primeiro e o quinto campo, funcionará da mesma forma que se ele preenchesse somente os dois primeiros.

Você pode alterar o script para aceitar mais ou menos de cinco arquivos, bastando adicionar ou retirar campos do tipo `file` no formulário e ajustar o valor da variável `$i` no script.

Utilizando um array

Essa forma de envio consiste em definir um único nome para todos os campos do tipo `file`, e esse

nome deve ser precedido pelos colchetes ([ ]).

Dessa forma, o script PHP irá considerar cada um desses campos como um elemento do array `$_FILES`, que ganhará uma terceira dimensão, representada pelo índice do arquivo. Isso pode lhe parecer confuso, mas na realidade esse método torna a programação muito mais simples. Veja a seguir, no arquivo que nomeamos como `upload_multiplo_array.html`, como ficará o formulário HTML:

upload_multiplo_array.html

```
<html>
<head>
<title>Livro de PHP - Upload de múltiplos arquivos</title>
</head>
<body>
<h2 align="center">Upload de múltiplos arquivos</h2>
<form method=POST action="executa_upload_multiplo_array.php"
    enctype=multipart/form-data>
<input type="hidden" name="MAX_FILE_SIZE" value="2000000">
<p align=center>Arquivo 1: <input type="file" name="arquivo[]" size="30"></p>
<p align=center>Arquivo 2: <input type="file" name="arquivo[]" size="30"></p>
<p align=center>Arquivo 3: <input type="file" name="arquivo[]" size="30"></p>
<p align=center>Arquivo 4: <input type="file" name="arquivo[]" size="30"></p>
<p align=center>Arquivo 5: <input type="file" name="arquivo[]" size="30"></p>
<p align=center><input type=submit value="Enviar arquivos"></p>
</form>
</body>
</html>
```

No programa que irá receber os arquivos, o `executa_upload_multiplo_array.php`, usamos o

comando de repetição for para acessar os elementos referentes a cada um dos arquivos recebidos. Em relação ao exemplo anterior, você poderá notar duas diferenças no tratamento dos arquivos. Acompanhe a seguir o código-fonte do programa e logo após a explicação:

executa_upload_multiplo_array.php

```
<html>
<head>
<title>Livro de PHP - Upload de múltiplos arquivos</title>
</head>
<body>
<h2 align="center">Upload de múltiplos arquivos</h2>
<?
// elimina o limite de tempo de execução
set_time_limit (0);
// inclui o arquivo com as configurações
include 'config_upload.inc';
// repete os mesmos comandos para os 5 arquivos
for ($i = 0; $i < 5; $i++) {
    $erro = FALSE;
    $nome_arquivo = $_FILES['arquivo']['name'][$i];
    $tamanho_arquivo = $_FILES['arquivo']['size'][$i];
    $arquivo_temporario = $_FILES['arquivo']['tmp_name'][$i];
    if (!empty ($nome_arquivo)) {
    if ($sobrescrever == "nao" && file_exists("$caminho_absoluto/$nome_arquivo")) {
        $erro = TRUE;
        echo "Arquivo $nome_arquivo já existe.";
           }
        if (($limitar_tamanho == "sim") && ($tamanho_arquivo > $tamanho_bytes)) {
```

```
            $erro = TRUE;
            echo "Arquivo $nome_arquivo deve ter no máximo $tamanho_bytes bytes.";
        }
        $ext = strrchr($nome_arquivo,'.');
        if ($limitar_ext == "sim" && !in_array($ext,$extensoes_validas)) {
            $erro = TRUE;
            echo "Extensão do arquivo $nome_arquivo inválida para upload.";
        }
        if (!$erro && move_uploaded_file($arquivo_temporario,
            "$caminho_absoluto/$nome_arquivo"))
            echo "<p align=center>0 upload do arquivo <b>$nome_arquivo</b>
                foi concluído com sucesso.</p>";
        else
            echo "<p align=center>0 arquivo $nome_arquivo não pôde ser copiado para o
                servidor.</p>";
    }
}
?>
</body>
</html>
```

Note que agora a variável de controle `$i` varia de 0 a 4, visto que o array tem início na posição zero. A outra diferença está no modo como são acessadas as informações sobre os arquivos, pois o array `$_FILES` ganhou uma terceira dimensão:

```
$nome_arquivo = $_FILES['arquivo']['name'][$i];
$tamanho_arquivo = $_FILES['arquivo']['size'][$i];
$arquivo_temporario = $_FILES['arquivo']['tmp_name'][$i];
```

A terceira dimensão representa o índice do arquivo no vetor. O primeiro arquivo enviado será o de

índice 0, o segundo será o de índice 1, e assim por diante. Nesse caso, usamos o valor da variável `$i` para representar esse índice. A programação torna-se mais simples, visto que não há necessidade de montar dinamicamente o identificador do arquivo. Assim como no exemplo anterior, você também pode alterar o script aumentando o intervalo de variação do `$i`, podendo permitir ao usuário enviar mais de cinco arquivos.

Problemas comuns

Caso você não consiga realizar a operação de upload, a seguir são listadas algumas das possíveis causas do problema:

- Verifique se o diretório no qual você está tentando gravar o arquivo possui permissão de escrita, caso contrário o PHP não conseguirá mover o arquivo do diretório temporário para o diretório destino. Por exemplo, no Linux, deve-se dar permissão de escrita ao diretório digitando `chmod 777 nome_diretorio`.
- O parâmetro `MAX_FILE_SIZE` do formulário HTML não pode especificar um tamanho de arquivo maior do que aquele especificado na diretiva `upload_max_filesize` do arquivo de configuração `php.ini`. O valor-padrão é 2 MB.
- Se o limite de memória estiver ativado (diretiva `memory_limit` do `php.ini`), poderá impedir o upload de seu arquivo. Certifique-se de que esse valor é grande o suficiente.
- Se a diretiva `max_execution_time` do `php.ini` possuir um valor muito pequeno, a execução de seu script poderá ultrapassar o limite de tempo e o upload será interrompido. Como vimos nos exemplos deste capítulo, se usarmos o comando `set_time_limit (0)` dentro do script, eliminaremos o limite de tempo e não precisaremos nos preocupar com a diretiva `max_execution_time`.

- A diretiva `post_max_size` do `php.ini` define o tamanho máximo para dados enviados pelo método `POST`. Se possuir um valor muito pequeno, isso poderá impedir o upload. Certifique-se de que esse valor é grande o suficiente.

Importante: por questões de segurança, faça sempre a validação do arquivo recebido no script PHP, principalmente quanto à extensão deste. Assim, você evita que o usuário possa enviar, por exemplo, um arquivo com extensão `.php` contendo um script que acessa dados confidenciais de seu sistema.

Mensagens de erro

Existe um elemento adicional no array `$_FILES`, que armazena um código de erro quando a operação de upload falhar. O nome da propriedade é `error`, ou seja, esse código pode ser acessado em `$_FILES['arquivo']['error']`, considerando que utilizamos o identificador "`arquivo`" no formulário HTML.

A seguir, serão apresentados os possíveis valores para esse elemento.

Constante	Descrição
`UPLOAD_ERR_OK`	Equivale ao valor 0. Indica que não houve erro e o upload ocorreu com sucesso.
`UPLOAD_ERR_INI_SIZE`	Equivale ao valor 1. Indica que o tamanho do arquivo recebido é maior que o valor definido na diretiva `upload_max_filesize` do `php.ini`.
`UPLOAD_ERR_FORM_SIZE`	Equivale ao valor 2. Indica que o tamanho do arquivo selecionado pelo usuário é maior que o valor definido no parâmetro `MAX_FILE_SIZE` do formulário HTML.
`UPLOAD_ERR_PARTIAL`	Equivale ao valor 3. Indica que o upload do arquivo foi feito parcialmente.
`UPLOAD_ERR_NO_FILE`	Equivale ao valor 4. Indica que o upload do arquivo não foi feito.

Funções de FTP disponíveis no PHP

Além das principais funções que vimos neste capítulo, existem várias funções adicionais para a utilização do protocolo FTP na linguagem PHP. A seguir, serão apresentadas as funções que podemos utilizar e uma pequena descrição de cada uma delas.

ftp_alloc

Aloca espaço para um arquivo a ser enviado para o servidor.

```
bool ftp_alloc (recurso id_conexão, int tamanho_arquivo,[string &resultado])
```

ftp_cdup

Altera para o diretório que está um nível acima (parent).

```
bool ftp_cdup (recurso id_conexão)
```

ftp_chdir

Altera o diretório em um servidor FTP.

```
bool ftp_chdir (recurso id_conexão, string diretório)
```

ftp_chmod

Define as permissões de um arquivo via FTP.

```
string ftp_chmod (recurso id_conexão, int modo, string nome_arquivo)
```

ftp_close

Fecha uma conexão FTP.

```
void ftp_close (recurso id_conexão)
```

ftp_connect

Abre uma conexão FTP.

```
recurso ftp_connect (string servidor [, int porta [, int tempo_max]])
```

ftp_delete

Exclui arquivos em um servidor FTP.

```
bool ftp_delete (recurso id_conexão, string caminho)
```

ftp_exec

Solicita a executação de um programa no servidor FTP.

```
bool ftp_exec (recurso id_conexão, string comando)
```

ftp_fget

Faz o download de um arquivo do servidor FTP e salva em um arquivo aberto.

```
bool ftp_fget (recurso id_conexão, recurso handle_arquivo, string arq_remoto,
    int modo [, int pos_reiniciar])
```

ftp_fput

Envia de um arquivo aberto para um servidor FTP.

```
bool ftp_fput (recurso id_conexão, string arq_remoto, recurso handle_arq,
    int modo [, int pos_inicial])
```

ftp_get_option

Retorna diversas configurações em tempo de execução da conexão FTP corrente.

```
misto ftp_get_option (recurso id_conexão, int opção)
```

ftp_get

Faz o download de um arquivo do servidor FTP.

```
bool ftp_get (recurso id_conexão, string arq_local, string arq_remoto, int modo
[, int pos_reiniciar])
```

ftp_login

Realiza o login em uma conexão FTP.

```
bool ftp_login (recurso id_conexão, string nome_usuário, string senha)
```

ftp_mdtm

Retorna a hora da última modificação de determinado arquivo.

```
int ftp_mdtm (recurso id_conexão, string arq_remoto)
```

ftp_mkdir

Cria um diretório.

```
string ftp_mkdir (recurso id_conexão, string diretório)
```

ftp_nb_continue

Continua enviando ou recebendo um arquivo (não bloqueante).

```
bool ftp_nb_continue (recurso id_conexão)
```

ftp_nb_fget

Retorna um arquivo de um servidor FTP e o grava em um arquivo aberto (não bloqueante).

```
bool ftp_nb_fget (recurso id_conexão, recurso handle_arq, string arq_remoto,
    int modo [, int pos_reiniciar])
```

ftp_nb_fput

Armazena um arquivo aberto em um servidor FTP (não bloqueante).

```
bool ftp_nb_fput (recurso id_conexão, string arq_remoto, recurso handle_arq,
    int modo [, int pos_inicial])
```

ftp_nb_get

Retorna um arquivo de um servidor FTP e o grava em um arquivo local (não bloqueante).

```
bool ftp_nb_get (recurso id_conexão, string arq_local, string arq_remoto,
    int modo [, int pos_reiniciar])
```

ftp_nb_put

Armazena um arquivo no servidor FTP (não bloqueante).

```
bool ftp_nb_put (recurso id_conexão, string arq_remoto, string arq_local,
    int modo [, int pos_inicial])
```

ftp_nlist

Retorna uma lista dos arquivos em determinado diretório.

```
array ftp_nlist (recurso id_conexão, string diretório)
```

ftp_pasv

Ativa ou desativa o modo passivo.

```
bool ftp_pasv (recurso id_conexão, bool passivo)
```

ftp_put

Envia (upload) um arquivo para um servidor FTP.

```
bool ftp_put (recurso id_conexão, string arq_remoto, string arq_local,
    int modo [, int pos_inicial])
```

ftp_pwd

Retorna o nome do diretório corrente.

```
string ftp_pwd (recurso id_conexão)
```

ftp_quit

Fecha uma conexão FTP. Apelido (alias) para a função `ftp_close`.

ftp_raw

Envia um comando arbitrário para um servidor FTP.

```
array ftp_raw (recurso id_conexão, string comando)
```

ftp_rawlist

Retorna uma lista detalhada dos arquivos em determinado diretório.

```
array ftp_rawlist (recurso id_conexão, string diretório)
```

ftp_rename

Renomeia arquivos em um servidor FTP.

```
bool ftp_rename (recurso id_conexão, string nome_atual, string novo_nome)
```

ftp_rmdir

Remove um diretório.

```
bool ftp_rmdir (recurso id_conexão, string diretório)
```

ftp_set_option

Define diversas opções do FTP em tempo de execução.

```
bool ftp_set_option (recurso id_conexão, int opção, misto valor)
```

ftp_site

Envia um comando do tipo SITE ao servidor.

```
bool ftp_site (recurso id_conexão, string comando)
```

ftp_size

Retorna o tamanho de determinado arquivo.

```
int ftp_size (recurso id_conexão, string arq_remoto)
```

ftp_ssl_connect

Abre uma conexão segura SSL-FTP.

```
recurso ftp_ssl_connect (string servidor [, int porta [, int tempo_max]])
```

ftp_systype

Retorna o identificador do tipo de sistema do servidor remoto de FTP.

```
string ftp_systype (recurso id_conexão)
```