WebPublish User's Manual

Documentation for WebPublish
Version 0.1.0.

Charles Henry Schoonover

Table of Contents

1 Introduction

WebPublish is a command line utility for the GNU operating system. WebPublish can be used to manage all of the details that are associated with publishing a local copy of a website to one or more remote servers using File Transfer Protocol (FTP). Using WebPublish to manage the transfer of a website allows a website developer to concentrate on a website's content.

The general way to use WebPublish is to create a website in your own local directory. Add, modify, and remove files and directories in the local copy of the website. After changes have been made to the local copy of the website, WebPublish can transfer the changes that you made to a remote server.

Using WebPublish, you can transfer only the files and directories that have been added or changed in the local copy of the website. Furthermore, WebPublish can remove the files and directories from the remote server that are no longer a part of the local copy of the website.

WebPublish provides a feature that allows you to exclude files and directories from being transferred to a server (e.g., a template directory or a project file). You can also exclude files and directories from being removed from a server (e.g. cgi-bin). WebPublish also allows you to associate a file transfer mode (text or binary) with a file extension.

Finally, a special feature of WebPublish allows you to have a shell program called each time that a file with a specific extension is about to be transferred to a remote server. This makes it easy to write a script or a program that can make alterations to a file just before it is transferred.

2 Installation

The source code for WebPublish is distributed as an archive file. The name of the archive file is '`webpublish-0.1.0.tar.gz`'. This compressed file contains the source code that must be compiled in order to build the WebPublish executable program. This section of the WebPublish manual explains how you can obtain the archive file and compile the program for yourself.

2.1 How to obtain WebPublish

WebPublish is a GNU software package. All GNU software packages are available from the GNU server (`www.gnu.org`). The home webpage for WebPublish is `www.gnu.org/software/webpublish`. Visit the WebPublish home webpage to download the WebPublish archive file.

2.2 Requirements

Except for the normal dependencies, such as a compiler and standard programming libraries, the WebPublish command line utility for GNU/Linux does not depend on any other packages.

2.3 Compilation and installation

The compressed WebPublish archive file must be un-compressed before you can compile the source code. The command to un-compress the archive file is the tar command. The specific command to un-compress the WebPublish archive file is:

```
tar zxvf webpublish-0.1.0.tar.gz
```

This command will extract the source code files from the archive file and store them in their own directory called '`./webpublish-0.1.0`'. In order to compile and install WebPublish on your system, change to the new directory '`./webpublish-0.1.0`' and type the following in the base directory of the WebPublish distribution:

```
./configure && make
```

Enter the *`su`* command to gain root privileges and enter the final command to install the software on your system.

```
make install
```

Since WebPublish uses autoconf you should not have trouble compiling it. If you have problems, please report them to the author at `bug-webpublish@gnu.org`.

3 Getting Started

WebPublish is a command line utility. Therefore, to use WebPublish, you must enter a string on the command line. The format for the string to use is:

```
webpublish COMMAND [[DATA]..] [[FLAG]..] [[FILE]..]
```

In this string, [COMMAND], page 23 represents the command that WebPublish will process. [DATA], page 23 represents the information that WebPublish needs in order to process the COMMAND. [FLAG], page 24 represents the optional flags that modify the behavior of WebPublish. And, finally, [FILE], page 24 represents individual file paths (and/or glob expressions).

3.1 Two modes of operation

WebPublish operates in two modes. The first mode allows you to manage the information in the WebPublish database. The second mode uses the information that is stored in the WebPublish database to publish or synchronize a website.

3.1.1 Managing the WebPublish database

When you execute WebPublish for the first time, WebPublish will create a hidden directory in your home directory. The hidden directory is named '`$HOME/.webpublish`'. WebPublish uses this hidden directory to store several database files.

There are four commands that are used to maintain the information in the WebPublish database. The four commands are:

'`[--add], page 27`'
: Add a new record to a database file.

'`[--change], page 27`'
: Change an existing record in a database file.

'`[--remove], page 27`'
: Remove an existing record from a database file.

'`[--list], page 27`'
: List one or all of the records in a database file.

There are five database files that the database commands can operate on. The five database files are:

'`account`' The account records are stored in this file.

'`shell`' The file alteration program records are stored in this file.

'`synch`' The synchronize path records are stored in this file.

'`publish`' The publish path records are stored in this file.

'`mode`' The file transfer mode records are stored in this file.

The [–file], page 27 ('`-F`') command line option is used to specify the database file that a given database command will operate on. For example, the following command string will add a new record to the account database file.

```
webpublish --add -F account (additional information)
```

3.1.2 Using the WebPublish database

WebPublish includes two commands that use the information in the WebPublish database. The two commands are [–publish], page 28 and [–synchronize], page 28. The '`--publish`' command uses the information in the WebPublish database to transfer files and directories to a remote server. The '`--synchronize`' command uses the information in the WebPublish database to remove files and directories from the remote server.

3.2 File and directory paths

Often, a file or a directory path is required on the command line. Sometimes the path will represent a file or a directory in the local copy of a website. Other times the path will represent a file or a directory on a server. Sometimes the path will be a complete path. Other times the path will be a relative path.

When a file or a directory path begins with the character '`/`', WebPublish assumes that the path is a complete path. When a complete path is passed to WebPublish, WebPublish will use the path without modification.

A complete path is never used when specifying a file or a directory on a server. Furthermore, when a complete path is used to specify a file or a directory in the local copy of a website, the complete path must be to a file or a directory that is in the local copy of the account's website. Specifying a file or a directory that is not in the account's website will cause an error.

When a file or a directory path does not begin with the character '`/`', WebPublish assumes that the path is a relative path. When a relative path is passed to WebPublish, WebPublish will resolve the path against the base address of the website. Either the base address of the local copy of a website, or the base address of the website on a server.

3.3 Modifying WebPublish behavior

Sometimes, it will be necessary to modify the behavior of WebPublish. For example, you may need descriptive messages written to stdout by WebPublish as WebPublish is performing an operation. Or, as another example, you may want WebPublish to ask you before WebPublish removes anything from the server. The FLAG command line options can get WebPublish to do these things and more.

The [FLAG], page 24 command line options can be used to modify the behavior of WebPublish. The FLAG options can be combined on the command line. The valid FLAG options are:

'`[--help], page 24`'
: Write a help message to stdout.

'`[--version], page 24`'
: Write a version message to stdout.

'`[--verbose], page 24`'
: Write descriptive messages to stdout.

'`[--force], page 24`'
: Force files to be transferred.

‘`[--prompt], page 25`’
: Prompt the user before removing anything from the server.

4 WebPublish Accounts

Before you can use WebPublish to publish a website, you must first create an account in the WebPublish database. An account contains the information that WebPublish uses to log into a remote server. A website can be transferred to more than one remote server by setting up an individual account for each server that you would like to transfer a website too.

4.1 Adding a new account

To create a WebPublish account, you must add a new record to the '`account`' database file. You can add a new record to the '`account`' database file by using the [–add], page 27 command with the following [DATA], page 23 options:

'`[--account], page 25`'
: The account name

'`[--website], page 25`'
: The local directory containing the website.

'`[--server], page 25`'
: The FTP server to transfer the website too.

'`[--user], page 25`'
: The user name to use when logging into the server.

'`[--password], page 26`'
: The password to use when logging into the server.

'`[--directory], page 26`'
: An optional subdirectory on the server to publish the website too.

The format of the command line string to use when adding a new record to the account database file is:

```
webpublish --add -F account -a ACCT -w SITE -s FTP
-u USER -p PASS [-d DIR]
```

ACCT should be replaced with the name that you would like to call the account. Some example account names are '`personal`', '`tripod`', '`weedguy`', and '`mywebsite`'. Whatever you decide to call the account, each account name must be unique.

SITE should be replaced with the local directory of the website that is managed by this account. This should be a full directory path. If, for example, you were creating an account for a website that was located at '`/home/mydir/mywebsite`' then you would replace *SITE* with the string '`/home/mydir/mywebsite`'.

FTP should be replaced with the name of the FTP server that WebPublish will transfer the website to. Some example FTP servers are '`ftp.tripod.com`', '`ftp.padresoftware.com`', and '`ftp.gnu.org`'.

USER should be replaced with the name that is used to log into the FTP server

PASS should be replaced with the password that is used to log into the FTP server.

[–directory], page 26 ('`-d`') is optional. When the '`--directory`' option is included on the command line, *DIR* should be replaced with the directory on the remote server

that the website is to be published too. For example, if you wish to publish a website to the directory '`photos/personal`' on a remote server then you would replace *DIR* with '`photos/personal`'. This will cause WebPublish to transfer the website to the '`photos/personal`' directory on the remote server. This option is especially useful when you publish more than one website to the same FTP server.

The following example will create a new WebPublish account called '`personal`' that will publish the website in the local directory '`/home/mydir/mysite`' to the FTP server `ftp.example.com`.

```
webpublish --add -F account -a personal -w /home/mydir/mysite
-s ftp.example.com -u myname -p mypass
```

4.2 Changing an existing account

To change an account's record, you must change the [DATA], page 23 in an existing record in the '`account`' database file. You can use the [–change], page 27 command line option to change the *DATA* in an account's record. The following *DATA* options can be used to change an account:

'`[--account], page 25`'
: The account name

'`[--website], page 25`'
: The local directory containing the website.

'`[--server], page 25`'
: The FTP server to transfer the website too.

'`[--user], page 25`'
: The user name to use when logging into the server.

'`[--password], page 26`'
: The password to use when logging into the server.

'`[--directory], page 26`'
: An optional subdirectory on the server to publish the website too.

To change one or more [DATA], page 23 items in an account's record, specify the name of the account to be changed, along with the *DATA* options that are to be changed. If a *DATA* option is specified on the command line then the *DATA* option's new value will overwrite the account's *DATA*. If a *DATA* option is not specified on the command line then the account's original *DATA* will remain unchanged. The only exception to this is the [–directory], page 26 *DATA* option. To clear the directory entry in an account's record, the '`--directory`' *DATA* option must contain the value '`/`'.

NOTE: You cannot change an account's name. If you need to change the name of an account, you must create a new account with the new name. Then you can remove the account with the old name.

The following example will change the user name and the password for the account '`personal`'. The example will also clear the '`--directory`' *DATA* item in the account's record.

```
webpublish --change -F account -a personal -u name -p pass -d /
```

4.3 Removing an existing account

To remove an existing account, you must remove the account's record from the '`account`' database file. You can use the [–remove], page 27 command line option to remove an existing account.

To remove an existing account from the WebPublish database, you must specify the name of the account that is to be removed. The [–account], page 25 ('`-a`') variable is used to specify the name of the account that is to be removed.

The following example will remove the account '`personal`' from the '`account`' database file.

```
webpublish --remove -F account -a personal
```

4.4 Listing accounts

The [–list], page 27 command can be used to list the information for a specific account to stdout. The '`--list`' command can also be used to list all of the accounts in the WebPublish database to stdout.

If an account name is specified on the command line using the [–account], page 25 ('`-a`') [DATA], page 23 option then only the specified account will be listed to stdout. If no account is specified on the command line then all of the accounts in the WebPublish database will be listed to stout.

The following example will list the account '`personal`'.

```
webpublish --list -F account -a personal
```

5 Excluding Publish Paths

WebPublish does not maintain a list of the files and directories that are a part of a website. WebPublish assumes that every file and subdirectory that is in a website's directory is a part of the website. When WebPublish publishes a website, WebPublish searches the website's directory to locate the files and directories that are to be transferred.

Often, the local copy of a website will contain files and directories that never need to be transferred to a server. Quanta, for example, will create a project file in a website's directory that never needs to be transferred to a server. A directory containing templates is an example of a directory that never needs to be transferred to a server. If you want to exclude a file or directory from being transferred to a server, you will need to add a record to the '`publish`' database file.

5.1 Adding an excluded publish path

To exclude a file or a directory from being published by WebPublish, you must add a record to the '`publish`' database file. To add a new record to the '`publish`' database file, you must specify the file or directory path using the [–path], page 26 (*-P*) [DATA], page 23 option.

The following example will prevent WebPublish from publishing the file '`website.project`' when WebPublish is publishing the personal account.

```
webpublish --add -F publish -a personal -P
/home/mydir/mysite/website.project
```

The excluded file or directory path must be a full path. Also, the excluded path is only valid for the specific account. When a website is published to more than one server, each account that is used to publish the website must have a record for the excluded path in the '`publish`' database file.

NOTE: The records in the '`publish`' database file cannot be changed. If you need to change a record in the '`publish`' database file, add a new record with the new information. Then you can remove the old record with the old information.

5.2 Removing an excluded publish path

To remove a file or directory path from the '`publish`' database file, you will need to specify the name of the account and the path that is to be removed.

The following example will remove the path for the file '`website.project`' from the '`publish`' database file.

```
webpublish --remove -F publish -a personal -P
/home/mydir/mysite/website.project
```

5.3 Listing excluded publish paths

The [–list], page 27 command can be used to list the excluded publish paths for a specific account to stdout. The '`--list`' command can also be used to list all of the excluded publish paths in the WebPublish database to stdout.

If an account name is specified on the command line using the [–account], page 25 (‘`-a`’) *DATA* option then only the excluded paths for the specified account will be listed to stdout. If no account is specified on the command line then all of the excluded paths in the WebPublish database will be listed to stout.

The following example will list all of the excluded publish paths for the account personal.

```
webpublish --list -F publish -a personal
```

6 Excluding Synch Paths

WebPublish does not maintain a list of the files and directories that have been transferred to a server. WebPublish assumes that every file and subdirectory in the server's directory is a part of the website. When WebPublish synchronizes a website, WebPublish searches the server's directory to locate the files and directories that are to be removed.

Often, the server's directory will contain files and directories that should never be removed from the server. '`cgi-bin`' is an example of a directory that should never be removed. If you want to exclude a file or directory from being removed from the server, you will need to add a record to the '`synch`' database file.

6.1 Adding an excluded synch path

To exclude a file or directory from being synchronized by WebPublish, you must add a record to the '`synch`' database file. To add a new record to the '`synch`' database file, you must specify the file or directory path using the [–path], page 26 ('`-P`') [DATA], page 23 option.

The following example will prevent WebPublish from synchronizing the directory '`cgi-bin`' when WebPublish is synchronizing the '`personal`' account.

```
webpublish --add -F synch -a personal -P cgi-bin
```

The excluded path is only valid for the specific account. When a website is published to more than one server, each account that is used to synchronize the website must have a record for the excluded path in the '`synch`' database file.

NOTE: The records in the '`synch`' database file cannot be changed. If you need to change a record in the '`synch`' database file, add a new record with the new information. Then you can remove the old record with the old information.

6.2 Removing an excluded synch path

To remove a file or directory path from the '`synch`' database file, you will need to specify the name of the account and the path that is to be removed.

The following example will remove the path for the directory '`cgi-bin`' from the '`synch`' database file.

```
webpublish -remove -F synch -a personal -P cgi-bin
```

6.3 Listing excluded synch paths

The [–list], page 27 command can be used to list the excluded synch paths for a specific account to stdout. The '`--list`' command can also be used to list all of the excluded synch paths in the WebPublish database to stdout.

If an account name is specified on the command line using the [–account], page 25 ('`-a`') [DATA], page 23 option then only the excluded paths for the specified account will be listed to stdout. If no account is specified on the command line then all of the excluded paths in the WebPublish database will be listed to stout.

The following example will list all of the excluded synch paths for the account personal.

```
webpublish -list -F synch -a personal
```

7 File Transfer Modes

WebPublish uses a very simple method for selecting the file transfer mode to use when transferring a file. WebPublish will use binary mode by default. If a shell program is called for the file just before the file is transferred then the file will be transferred in text mode.

If you would like to override the default file transfer mode then you will need to add a record to the '`mode`' database file. If a file's extension is listed in the '`mode`' database file then the file will be transferred using the extension's associated file transfer mode.

You do not need to specify the name of an account when you add a record to the '`mode`' database file. When you override the default file transfer mode, the new mode will be used by all accounts. This means that if the '`css`' file extension is listed in the '`mode`' database file then all '`css`' files that are transferred by WebPublish will be transferred using the '`css`' file transfer mode.

7.1 Adding a file transfer mode

To add a new file transfer mode to the WebPublish database, you must add a new record to the '`mode`' database file. To add a new record to the '`mode`' database file, you must specify a file extension and a file transfer mode.

The following example will override the file transfer mode for all of the files that are transferred with a '`css`' file extension. All files that have a '`css`' file extension will be transferred in text mode.

```
webpublish --add -F mode -e css -m text
```

7.2 Changing a file transfer mode

To change a file transfer mode, you must change a record in the '`mode`' database file. To change a record in the '`mode`' database file, you must specify the file extension to be changed and the new file transfer mode.

The following example will change the file transfer mode that is used for all of the files that are transferred with a '`css`' file extension. All files that have a '`css`' file extension will be transferred in binary mode.

```
webpublish --change -F mode -e css -m binary
```

7.3 Removing a file transfer mode

To remove a file transfer mode from the WebPublish database, you must remove a record from the '`mode`' database file. To remove a record from the '`mode`' database file, you must specify the file extension that is to be removed.

The following example will remove the file transfer mode that is used by all of the files that are transferred with the '`css`' file extension.

```
webpublish --remove -F mode -e css
```

7.4 Listing file transfer modes

The [–list], page 27 command can be used to list a specific file transfer mode to stdout. The '`--list`' command can also be used to list all of the file transfer modes in the WebPublish database to stdout.

If a file extension is specified on the command line using the [–extension], page 26 ('`-e`') [DATA], page 23 option then only the file transfer mode for the specified extension will be listed to stdout. If no file extension is specified on the command line then all of the file transfer modes in the WebPublish database will be listed to stout.

The following example will list the file transfer mode that is used for all of the files that have a '`css`' extension.

```
webpublish --list -F mode -e css
```

8 Making Alterations

WebPublish is generally used to publish a local copy of a website to one or more remote servers. Often, it is necessary to make changes to a file when it is transferred to a remote server. An HTML file, for example, may need to have a time stamp added. Also, files that use a <BASE> tag should have the base directory changed to work correctly from the server. WebPublish includes a feature that allows a shell program to be called just before a file with an associated file extension is about to be transferred. This makes it easy to write a script or a program that can alter a file just before it is transferred.

The '`shell`' database file contains a list of file extensions. Each file extension that is listed in the '`shell`' database file has a shell program associated with it. Just before WebPublish transfers a file to a remote server, WebPublish will search the '`shell`' database file. If the extension for the file that is being transferred matches a record in the '`shell`' database file then WebPublish will call the shell program that is associated with the file extension. The shell program can then change the file that is about to be transferred.

Each association is specific to each WebPublish account. That is, if you associate a shell program with the '`html`' file extension using the '`weedguy`' account then the shell program will only be called when you publish the '`weedguy`' account. If you want a shell program to be called with other accounts then you will have to associate the shell program with an extension for each account.

Each time that WebPublish executes a shell program, WebPublish will include the account information on the shell program command line. The account information that is passed to the shell program as command line arguments are:

'`-a ACCT`' *ACCT* specifies the name of the account that is being published.

'`-w DIR`' *DIR* specifies the local base directory of the website that is being published.

'`-s FTP`' *FTP* specifies the name of the server that the file is being transferred too.

'`-u USER`' *USER* specifies the user name that was used to log into the FTP server.

'`-e EXT`' *EXT* specifies the extension of the file that is being transferred.

'`-d DIR`' If this argument is included on the command line then *DIR* specifies the offset directory on the server for the account that is being published.

The shell program can read in the file from stdin and write the file to stdout. If the shell program does not make any changes to the file then the shell program can return 2. If the shell program returns a value of 2 then WebPublish will transfer the original file. If the shell program makes changes to the file then the shell program can return 3. If the shell program returns a value of 3 then WebPublish will transfer the file that was written to stdout by the shell program. If the shell program encounters an error then the shell program can return 1. If the shell program returns a value of 1 then WebPublish will report an error and terminate.

NOTE: stdout is piped to a temporary file in the WebPublish working directory ('`$HOME/.webpublish`'). Therefore, the local copy of the website file is never altered by the alteration function.

8.1 Adding a shell program

To associate a shell program with a file extension for a given account, you must add a record to the '`shell`' database file. To add a record to the '`shell`' database file, you must specify the file extension and the full path of the shell program.

The following example will associate the '`changebase`' shell program with the '`html`' file extension for the '`personal`' account.

```
webpublish --add -F shell -a personal -e html -P /home/mydir/changebase
```

This example will cause the '`changebase`' program to be called everytime that WebPublish is about to transfer a file with an '`html`' file extension, while publishing the '`personal`' account.

8.2 Changing a shell program

To change a shell program for an associated extension, you need to change the record in the '`shell`' database file. To change the record in the '`shell`' database file, you must specify the new value for the shell program.

The following example will change the path for the changebase shell program to '`/home/support/changebase`'.

```
webpublish --change -F shell -a personal -e html -P /home/support/changebase
```

8.3 Removing a shell program

To remove a shell program from the '`shell`' database file, you must specify the file extension that is to be removed.

The following example will remove the shell program that is associated with the '`html`' file extension.

```
webpublish -remove -F shell -a personal -e html
```

8.4 Listing shell programs

The [–list], page 27 command can be used to list the shell program for a specific account to stdout. The '`--list`' command can also be used to list all of the shell programs in the WebPublish database to stdout.

If an account name is specified on the command line using the [–account], page 25 ('`-a`') [DATA], page 23 option then only the shell programs for the specified account will be listed to stdout. If no account is specified on the command line then all of the shell programs in the WebPublish database will be listed to stout.

The following example will list all of the shell programs for the '`personal`' account.

```
webpublish --list -F shell -a personal
```

9 Publishing a Website

WebPublish is designed to manage the details associated with publishing a website from a local directory to a remote server using File Transfer Protocol (FTP). This allows a website developer to work on the local copy of a website and then use WebPublish to mirror the changes to one or more remote servers.

When WebPublish selects a file for publication, it does not necessarily mean that the file will be transferred. The file will only be transferred to the remote server if one of the following conditions is true: The [–force], page 24 (‘`-f`’) [FLAG], page 24 was included on the command line; the local copy of the file does not exist on the remote server; or the local copy of the file is newer than the copy on the remote server.

NOTE: Because the system time is not the same on each FTP server, WebPublish may continue to transfer files to the remote server even though the local copy of the file is not newer than the copy on the server, or WebPublish may not transfer files that should be transferred to the server. As the time difference should never be more than a day, this should not be a problem if you only publish a website one or less times a day. A future version of WebPublish will attempt to correct this problem by autodetecting the time difference and saving the difference in the account’s database record.

WebPublish needs to know which files should be published. There are three ways to tell WebPublish which files should be published. The first method is to specify each file on the command line. The second method is to include qualifiers on the command line. The third method is to use the ‘`touch`’ and ‘`find`’ commands.

9.1 Specifying files on the command Line

If you are interested in publishing specific account files then you can specify the individual files on the WebPublish command line. For example, if you have an account set up called ‘`personal`’ and your computer automatically updates the home webpage from time to time, you can tell WebPublish to publish your updated home page by specifying the file on the command line.

```
webpublish --publish -a personal home.html
```

The example above will publish the file ‘`home.html`’ to the ‘`personal`’ account’s server. Of course, the file ‘`home.html`’ must be located in the base directory of the account’s website.

Several files can be included on the command line. For example:

```
webpublish --publish -a personal home.html index.html etc/about.html media/photo.jp
```

The example above will publish the files ‘`home.html`’, ‘`index.hmtl`’, ‘`etc/about.html`’, and ‘`media/photo.jpg`’ to the ‘`personal`’ account’s server.

It is also correct to include the full file path of the individual files. The following example will publish the ‘`home.html`’ file to the ‘`personal`’ account’s server.

```
webpublish --publish -a personal /home/mydir/mysite/home.html
```

The full file path must be included if you would like to use wildcards to select files. For example, if you would like to publish all of the ‘`jpg`’ files in the ‘`etc`’ directory of the ‘`personal`’ account’s website then you can use the following example.

```
webpublish --publish -a personal /home/mydir/mysite/etc/*.jpg
```

9.2 Publish qualifiers

Qualifiers tell WebPublish how to select files for publication. Qualifiers are simply WebPublish [DATA], page 23 options that contain information that can be used to select files. If qualifiers are included on the command line then WebPublish will use the specified qualifiers to select the individual files for publication. If no qualifiers are included on the command line then everything will be selected.

The legal qualifiers that can be used with the [–publish], page 28 command are: [–extension], page 26, [–directory], page 26, and [–depth], page 26. The '`--extension`' qualifier will tell WebPublish to select files with a specified file extension. The '`--directory`' qualifier will tell WebPublish the directory to begin publishing. The '`--depth`' qualifier tells WebPublish how many directory levels to traverse. Qualifiers can be used individually or in combination.

9.3 Using publishing qualifiers

The simpliest way to use WebPublish is to let the utility select the entire website for publication. When WebPublish selects the entire website for publication, each file in the local copy of the website is examined to see if it is current on the remote server. Any files and directories that are not current on the server will be transferred by WebPublish. The following example will select the entire '`personal`' account's website for publication.

```
webpublish --publish -a personal
```

If you only want to publish '`jpg`' files, then you can use the [–extension], page 26 ('`-e`') qualifier. The '`--extension`' qualifier will tell WebPublish to select the files with the specified file extension. The following example will publish all of the '`jpg`' files in the '`personal`' account.

```
webpublish --publish -a personal -e jpg
```

The [–directory], page 26 ('`-d`') qualifier can be used to select a subdirectory in the account's website to publish. The specified directory and all of its subdirectories will be selected for publication. The following example will publish the '`books/biography`' directory in the '`personal`' account.

```
webpublish --publish -a personal -d books/biography
```

The [–depth], page 26 ('`-D`') qualifier can be used to limit the number of subdirectories that are published. If you only want to publish the files that are in the books directory, for example, you can use the following:

```
webpublish --publish -a personal -D 1 -d books
```

The example above will tell WebPublish to publish the single directory level that begins at the directory '`books`'.

9.4 Using touch and find

The '`touch`' and the '`find`' command line utilities can be used to select the files that should be transferred to a server. The '`touch`' command can be used to set a file's modification time to the current time. The '`find`' command can be used to select the files that have their modification time set to the current time. Setting a file's modification time to

the current time will cause WebPublish to transfer the file when WebPublish is publishing an account.

The '`touch`' command can be used to set a file's modification time to the current time. If the '`touch`' command is used on a file that is a part of a website, the file will be more recent than the copy of the file that is on the server. When WebPublish is publishing the account, WebPublish will transfer the file. If, for example, you want to force WebPublish to transfer a website's index file, you can use the '`touch`' command.

```
touch index.html
```

The example above will set the '`index.html`' file's modification time to the current time. When the website is published by WebPublish, the '`index.html`' file will be transferred to the server.

The '`find`' command can be used with the '`touch`' command to touch multiple website files. This will cause WebPublish to transfer all of the files that were selected with the '`find`' command. If, for example, you would like to transfer all of the files with a '`jpg`' file extension, you can use the '`find`' command with the '`-exec`' option.

```
find -name "*.jpg" -exec touch {} \;
```

The example above will use the '`-exec`' option of the '`find`' command to '`touch`' all of the files in the website that have a '`jpg`' file extension. This will cause WebPublish to transfer all of the '`jpg`' files when the website is being published by WebPublish.

10 Synchronizing a Website

The [–synchronize], page 28 command can be used to remove files and directories from the server that are no longer a part of the local copy of an account's website.

10.1 Synchronize qualifiers

Qualifiers tell WebPublish how to select files for synchronization. Qualifiers are simply WebPublish [DATA], page 23 options that contain information that can be used to select files. If qualifiers are included on the command line then WebPublish will use the specified qualifiers to select the individual files for synchronization. If no qualifiers are included on the command line then everything will be synchronized on the remote server.

The legal qualifiers that can be used with the [–synchronize], page 28 command are: [–directory], page 26 and [–depth], page 26. The '`--directory`' qualifier tells WebPublish the subdirectory to synchronize. The '`--depth`' qualifier tells WebPublish how many directory levels to traverse. Qualifiers can be used individually or in combination.

10.2 Using synchronizing qualifiers

The simplest way to use the [–synchronize], page 28 command is to synchronize the entire website. This will cause WebPublish to search the copy of the website on the server, looking for files and directories that are no longer a part of the local copy of the website. When WebPublish finds a file or a directory on the server that is no longer a part of the local copy of the website, WebPublish will remove the file or directory from the server (unless the [–prompt], page 25 switch was included on the command line).

The following example will synchronize the entire website for the '`personal`' account.

```
webpublish --synchronize -a personal --prompt
```

If you want to synchronize part of the website on a server then you can tell WebPublish the specific subdirectory to synchronize.

```
webpublish --synchronize -a personal -d books
```

The example above will only synchronize the books directory. No other part of the website on the server will be synchronized. Any files and directories in the books directory that are no longer a part of the books directory in the local copy of the website will be removed.

If you only want to synchronize the individual directory, you can tell WebPublish how many directory levels too traverse.

```
webpublish --synchronize -a personal -d books -D 1
```

The example above will only synchronize the '`books`' directory. WebPublish will not synchronize any subdirectories. If the [–depth], page 26 ('`-D`') [DATA], page 23 option had been 2, one single level of each subdirectory in the '`books`' directory would have been synchronized. Etc.

11 A Trivial Example

The advantage that WebPublish has over other website publishers is its ability to call a script or a program just before a file is transferred to a server (see Chapter 8 [Making Alterations], page 14). The script or program can add a time stamp to a file, change a file's <BASE> tag address, include common HTML sections to a webpage, etc.

This chapter will present a trivial example that will demonstrate how you can use WebPublish to develop and maintain a website. This example will create a website in a local directory. The example website will be published by WebPublish to a remote server. When files are transferred to the server, the base address of all '`html`' and RealMedia '`ram`' files will be changed to work correctly from the server. This development scheme makes it possible to develop a new section for a website locally. The local base address in the '`html`' and RealMedia '`ram`' files will prevent the additions from contaminating the live website on the server. When the new section is developed, WebPublish can transfer the additions to the server.

In our example, Charles, our website developer, will set up his own personal website. Charles will create and maintain his website in the local directory '`/mnt/linux/websites/personal`'. Charles will publish this website to the FTP server `ftp.xanadu2.net`. Publishing the website to `ftp.xanadu2.net` will make the website available to the public over the internet.

First, Charles creates the website directory. Charles also creates a subdirectory that will hold the website's templates. Finally, Charles writes a simple index webpage that explains that the website is under construction. The index webpage contains a local base address (<BASE href="file:/mnt/linux/websites/personal/">).

Here is a directory listing of the new website in the local directory.

```
[charles@localhost personal]$ ls -Fl
total 8
-rw-r--r--  1 charles  charles      0137 Oct 28 08:16 index.html
drwxr-xr-x  2 charles  charles      4096 Oct 28 08:15 templates/
```

Next, Charles sets up the WebPublish account that will publish his personal website to the FTP server. In this example, Charles creates a WebPublish account called, '`personal`'.

```
webpublish --add -F account -a personal -w
/mnt/linux/websites/personal -s ftp.xanadu2.net -u charles -p
example
```

The example above creates the '`personal`' WebPublish account. The '`personal`' account can be used to publish the website that is located at '`/mnt/linux/websites/personal`' to the FTP server `ftp.xanadu2.net`. WebPublish will log into the FTP server using the specified user name ([–user], page 25) and the password ([–password], page 26). Here is a listing of the account.

```
[charles@localhost personal]$ webpublish --list -F account -a
personal
Listing WebPublish account:
Account  : personal
Website  : /mnt/linux/websites/personal
Server   : ftp.xanadu2.net
```

```
User     : charles
Password : example
Directory:
```

After Charles creates the '`personal`' WebPublish account, Charles excludes the '`template`' directory from being transferred to the server by WebPublish.

```
webpublish --add -F publish -a personal -P
/mnt/linux/websites/personal/templates
```

The example above adds the path '`/mnt/linux/websites/personal/templates`' to the '`publish`' database file. This will cause WebPublish to ignore the '`template`' directory when WebPublish is searching for files and directories that need to be transferred to the server while the '`personal`' account is being published. Here is the listing of the record.

```
[charles@localhost personal]$ webpublish --list -F publish -a
personal
Listing local paths for personal:

Path: /mnt/linux/websites/personal/templates
```

Next, Charles excludes the server directory '`cgi-bin`' from being removed by WebPublish when WebPublish synchronizes the account.

```
webpublish --add -F -synch -a personal -P cgi-bin
```

The example above will add the path '`cgi-bin`' to the '`synch`' database file. This will cause WebPublish to ignore the '`cgi-bin`' directory when WebPublish is searching for files and directories that need to be removed from the server while the '`personal`' account is being synchronized. Here is the listing of the record.

```
[charles@localhost personal]$ webpublish --list -F synch -a
personal
Listing server paths for personal:

Path: cgi-bin
```

Next, Charles will associate an alteration program with the '`html`' file extension and the RealMedia '`ram`' file extension. This will cause the alteration program to be called just before a file with an '`html`' extension or a RealMedia '`ram`' extension is transferred to the server.

```
webpublish --add -F shell -a personal -e html -P
/mnt/linux/websites/support/changebase
webpublish --add -F shell -a personal -e ram -P
/mnt/linux/websites/support/changebase
```

The example above will add two records to the '`shell`' database file. The two records associate the '`html`' and RealMedia '`ram`' file extensions with the program '`changebase`' that is located in the directory '`/mnt/linux/websites/support`'. This will cause WebPublish to call the '`changebase`' program just before a file with an '`html`' or RealMedia '`ram`' file extension is about to be transferred while the '`personal`' account is being published. If the '`changebase`' program changes the base address in a file then the altered file will be transferred to the server by WebPublish. Here is a listing of the records.

```
[charles@localhost personal]$ webpublish --list -F shell -a
personal
```

```
Listing shell programs for personal:

Extension: html
Program  : /mnt/linux/websites/support/changebase

Extension: ram
Program  : /mnt/linux/websites/support/changebase
```

NOTE: Appendix A [The changebase Source Code], page 29 includes the source code for the '`changebase`' program. The program is a simple C++ program that can be used to change the base address of '`html`' and RealMedia '`ram`' files. The program also demonstrates how to use one program to manage multiple websites.

The last thing that Charles needs to do before he can concentrate fully on the development of the website's content is to make sure that all of the file transfer modes are correct. In our example, Charles needs to make sure that all files with a '`css`' extension (the style sheets) are transferred in '`text`' mode.

```
webpublish --add -F mode -e css -m text
```

The example above will add the '`css`' file extension to the '`mode`' database file. This will cause WebPublish to transfer all files with a '`css`' file extension to the server using '`text`' mode. Here is a listing of the record.

```
[charles@localhost personal]$ webpublish --list -F mode
Listing all file transfer modes:

Extension: css
Mode     : TEXT
```

Charles is now ready to publish his website to the server for the first time.

```
webpublish -publish -a personal -V
```

The example above will publish the personal account. In this case, WebPublish will transfer the file '`index.html`' to the server. Since the file has an '`html`' extension, WebPublish will call the '`changebase`' program just before the file is transferred. The '`index.html`' file has a base address of '`<BASE href="file:/mnt/linux/websites/personal/">`'. The base address will be changed by the '`changebase`' program to '`<BASE href="charles.padresoftware.com/">`'. This will cause the base address to work correctly from the server. (The [–version], page 24 [FLAG], page 24 causes WebPublish to write messages to stdout that describe what WebPublish is doing as WebPublish does it.) Since the '`templates`' directory is excluded, WebPublish will ignore it.

Charles can now concentrate on developing the content for his new website. Thanks to WebPublish, Charles does not have to spend any more time keeping track of the files and directories that need to be transferred to the server. WebPublish will know when files and directories have been added or changed and only the files and directories that have been added or changed will be transferred to the server.

This development scheme allows Charles to work on his home webpage without contaminating the website that is available to the public on the server.

Charles is really smart for using GNU/Linux and WebPublish.

12 Reference

WebPublish transfers a local copy of a website to a remote FTP server using an account. An account contains the information that WebPublish uses to transfer a website to an FTP server.

Usage: webpublish COMMAND [[DATA]...] [[FLAG]...] [[FILE]...]

COMMAND is always required on the command line (except when the [–help], page 24, and/or [–version], page 24, FLAG is included). Only one *COMMAND* can be included on each command line. The *COMMAND* tells WebPublish what to do. The valid *COMMAND*s are:

‘`--publish`’
: Publish files and directories to server.

‘`--synchronize`’
: Remove files and directories from server.

‘`--add`’ Add a new database record.

‘`--change`’
: Change an existing database record.

‘`--remove`’
: Remove an existing database record.

‘`--list`’ List database records.

DATA command line options specify the information that WebPublish needs to perform the requested *COMMAND*. Most *COMMAND*s require more than one *DATA* option to be included on the command line. Each *DATA* option requires an argument. The *DATA* options are:

‘`-a or --account ACCT`’
: *ACCT* specifies the name of the account to operate on.

‘`-w or --website DIR`’
: *DIR* specifies the website (the local directory) to operate on.

‘`-s or --server FTP`’
: *FTP* specifies the name of the FTP server to connect with.

‘`-u or --user USER`’
: *USER* specifies the user name to use when logging into the FTP server.

‘`-p or --password PASS`’
: *PASS* specifies the password to use when logging into the FTP server.

‘`-d or --directory DIR`’
: *DIR* specifies a directory path.

‘`-e or --extension EXT`’
: *EXT* specifies a file extension.

‘`-p or --path PATH`’
: *PATH* specifies a file or directory path.

'`-D or --depth NUM`'
: *NUM* specifies the number of directory levels to traverse.

'`-F or -file FILE`'
: *FILE* specifies the database file to operate on. The value of *FILE* can be either '`account`', '`shell`', '`synch`', '`publish`', or '`mode`' (case is not compared).

The WebPublish *FLAG*s modify the behavior of WebPublish. Multiple *FLAG*s are allowed on the same command line. The valid *FLAG*s are:

'`-h or --help`'
: Write this help message to stdout.

'`-v or --version`'
: Write a version string to stdout.

'`-V or --verbose`'
: Write descriptive messages to stdout.

'`-f or --force`'
: Force files to be transferred.

'`--prompt`'
: Prompt the user before removing anythingfrom the server.

Specific *FILE*s can be published by listing each file on the command line. The file path can be a full path or a relative path. Wildcards can be used with a full path.

The following example will create a new account called '`example`'. The account will publish the website that is located at '`/home/mydir/example`' to the ftp server `ftp.example.com`.

```
webpublish --add -F account -a example -s ftp.example.com -u
testguy -p testpass -w /home/mydir/example
```

The following example will publish the account '`example`'.

```
webpublish --publish -a example -V
```

12.1 -h or –help

This command line switch will cause WebPublish to write a help message to stdout. The help message will include a brief description of each WebPublish command line option.

12.2 -v or –version

This command line switch will cause WebPublish to write a brief version message to stdout.

12.3 -V or –verbose

This command line switch will cause WebPublish to write a lot of descriptive messages to stdout. The messages will describe what WebPublish is doing, as WebPublish does it.

12.4 -f or –force

This command line switch will cause WebPublish to transfer all selected files. Normally, WebPublish will only transfer the selected files that do not already exist on the server or that have a file modification date on the server that is older than the local copy of the file.

12.5 –prompt

This command line switch will cause WebPublish to prompt the user before WebPublish removes a file or directory from the server. When the user is prompted, the user will be given the choice of answering *y* for '`yes`', *n* for '`no`', or `a` for '`add`'. If the user answers *y* then the file or directory will be removed from the server. If the user answers *n* then the file or directory will not be removed from the server. If the user answers `a` then the file or directory will not be removed from the server and the path for the file or directory will be added to the '`synch`' database file. This will cause WebPublish to exclude the file or directory from being synchronized in the future.

12.6 -a or –account

This is the command line option that is to be used when you must include an account name. An account name is required by most WebPublish command line actions. Some examples of an account name are:

```
-a personal
-a business
-a photowebsite
```

12.7 -w or –website

This is the command line option to use when you must specify a website on the command line. The value for this variable must be the complete path to the local directory that contains the website. An example website is:

```
-w /home/mydir/mywebsite
```

12.8 -s or –server

This is the command line option that is to be used when you must specify the name of an FTP server on the command line. The value for this variable must be the name of the FTP server. Some example server names are:

```
-s localhost.localdomain
-s ftp.tripod.com
-s ftp.padresoftware.com
```

12.9 -u or –user

This is the command line option that is to be used when you must specify the name to use when logging into the FTP server. Some example user names are:

```
-u weedguy
-u charles
-u bigstud
```

12.10 -p or –password

This is the command line option that is to be used when you must specify the password to use when logging into the FTP server. Some example passwords are:

```
-p f9f012j2
-p frogman
-p bigstud
```

12.11 -d or –directory

This is the command line option that is to be used when you must specify a directory on the command line. If a directory path begins with '/' then the path is assumed to be a full path. Otherwise, the path is assumed to be relative to the base directory of the website. Either the base address of the local copy of the website, or the base address of the website on the server.

Some example directories are:

```
-d /home/mydir/mywebsite
-d cgi-bin
-d photos/party
```

12.12 -e or –extension

This is the command line option that is to be used when you must specify a file extension on the command line. Some example file extensions are:

```
-e html
-e ram
-e jpg
```

12.13 -P or –path

This is the command line option that is to be used when you must specify a complete path to a file or to a directory. If a path begins with '/' then the path is assumed to be a full path. Otherwise, the path is assumed to be relative to the base directory of the website. Either the base address of the local copy of the website, or the base address of the website on the server. Some example paths are:

```
-P /home/mydir/mywebsite/index.html
-P cgi-bin
-P index.html
```

12.14 -D or –depth

This is the command line option that is to be used when you must specify the maximum number of directory levels to traverse when publishing or synchronizing a website. Some example directory levels are:

```
-D 1
-D 2
-D 5
```

12.15 -F or –file

This is the command line option that is to be used when you must specify a database file to operate on. The value of [–file], page 27 can be either '`account`', '`shell`', '`synch`', '`publish`', or '`mode`'. Here is an example that will add a record to the mode database file.

```
webpublish -add -F mode -e html -m text
```

12.16 –add

This is the command line option that is to be used when you must add a record to a WebPublish database file. The [–file], page 27 *DATA* option determines which file will be operated on. The value of [–file], page 27 can be either '`account`', '`shell`', '`synch`', '`publish`', or '`mode`'. Here is an example that will add a record to the '`mode`' database file.

```
webpublish --add -F mode -e html -m text
```

12.17 –change

This is the command line option that is to be used when you must change a record in a WebPublish database file. The [–file], page 27 *DATA* option determines which file will be operated on. The value of [–file], page 27 can be either '`account`', '`shell`', '`synch`', '`publish`', or '`mode`'. Data that is included on the command line will overwrite the original data values. Data that is not included on the command line will remain unchanged in the record. Here is an example that will change a record in the '`mode`' database file.

```
webpublish --change -F mode -e css -m text
```

12.18 –remove

This is the command line option that is to be used when you must remove a record from a WebPublish database file. The [–file], page 27 *DATA* option determines which file will be operated on. The value of [–file], page 27 can be either '`account`', '`shell`', '`synch`', '`publish`', or '`mode`'. Here is an example that will remove a record from the mode database file.

```
webpublish --remove -F mode -e html
```

12.19 –list

This is the command line option that is to be used when you must list one or more records in a WebPublish database file to stdout. The [–file], page 27 *DATA* option determines which file will be operated on. The value of [–file], page 27 can be either '`account`', '`shell`', '`synch`', '`publish`', or '`mode`'. If an account name is specified on the command line then only the items for the specified account will be listed. If an account name is not specified on the command line then all of the items in the WebPublish database file will be listed to stdout. Here is an example that will list all of the accounts in the WebPublish database.

```
webpublish --list -F account
```

12.20 –publish

This is the command line option that is to be used when you want to transfer files and directories to a server. See Chapter 9 [Publishing a Website], page 16, for a detailed description on the use of this command line option.

12.21 –synchronize

This is the command line option that is to be used when you want to remove files and directories from a server. See Chapter 10 [Synchronizing a Website], page 19, for a detailed description on the use of this command line option.

Appendix A The changebase Source Code

This appendix contains the source code for the changebase program. The changebase program is used by Padre Software to alter '`html`' and RealMedia '`ram`' files just before they are transferred by WebPublish. Specifically, the changebase program will change the local base directory that is specified in the '`html`' and '`ram`' files to the base directory that it should be on the remote server.

The function '`parse_command_line()`' is called to read in the command line arguments that were passed to the program by WebPublish. The command line arguments contain the account information for the website that is being published.

The function '`change_html()`' will alter the base address in an '`html`' file to the address that it should be on the server. For example, a local base address of '`<BASE href="file:/mnt/linux/websites/weedguy/">`', will be changed to '`<BASE href="http://weedguy.padresoftware.com/">`'.

The function '`change_ram()`' will alter the base address in a RealMedia '`ram`' file to the address that it should be on the server. For example, a local base address of '`file:/mnt/linux/websites/weedguy/`', will be changed to '`http://weedguy.padresoftware.com/`'.

Finally, the main program function will use the information that was included on the command line to determine which website is being published and which base address is to be used when changing a base address.

```
// Sample HTML and RAM alteration program for WebPublish 0.0.0

#include <iostream.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <getopt.h>             // 'getopt_long' function.

#define OK                      0
#define ERROR                   1
#define NOCHANGES               2
#define CHANGES                 3

char*           account         = (char*)0;
char*           website         = (char*)0;
char*           server          = (char*)0;
char*           user            = (char*)0;
char*           directory       = (char*)0;
char*           extension       = (char*)0;

static char shortoptions[]      = "a:w:s:u:d:e:";
static struct option            longoptions[] =
   {
      { "account",     true,   0,              'a'             },
      { "website",     true,   0,              'w'             },
      { "server",      true,   0,              's'             },
```

```
      { "user",        true,   0,              'u'             },
      { "directory",   true,   0,              'd'             },
      { "extension",   true,   0,              'e'             },

      { 0,             0,      0,               0              }
   };

int store_parse_option(char*& item, char *optarg)
   {
      int               result          = OK;

      if (item != (char*)0)
         {
            cerr << "Item already allocated.\n";
            result      = ERROR;
         }
      else
         {
            item        = (char*)malloc(strlen(optarg) + 1);
            if (item == (char*)0)
               {
                  cerr << "Could not allocate memory.\n";
                  result        = ERROR;
               }
            else
               {
                  strcpy(item, optarg);
               }
         }
      return(result);
   }

// Parse the command line options.

int parse_command_line(int argc, char** argv)
   {
      int               result          = OK;
      int               option;                 // Switch value.
      int               flagindex       = 0;    // Cmd line flag index.

      while ((option = getopt_long(argc, argv, shortoptions, longoptions,
         &flagindex)) != EOF && result == OK)
         {
            switch(option)
               {
                  case 'a':             // -a or --account
                     result     = store_parse_option(
                                  account, optarg);
                     break;
```

```
                    case 'w':               // -w or --website
                       result     = store_parse_option(
                                    website, optarg);
                       break;
                    case 's':               // -s or --server
                       result     = store_parse_option(
                                    server, optarg);
                       break;
                    case 'u':               // -u or --user
                       result     = store_parse_option(
                                    user, optarg);
                       break;
                    case 'd':               // -d or --directory
                       result     = store_parse_option(
                                    directory, optarg);
                       break;
                    case 'e':               // -e or --extension
                       result     = store_parse_option(
                                    extension, optarg);
                       break;
                    default:                // Unsupported
                       result     = ERROR;
                       cerr << "Unknown command line option.\n";
                       break;
                 }
              }
           return(result);
        }

     int change_html(const char* base)
        {
           int               result          = NOCHANGES;
           char*             pointer;
           char              buffer[1024];

           while (fgets(buffer, 1024, stdin) != 0)
              {
                 if ((pointer = strstr(buffer, "\n", buffer, base);
                    {
                       result        = CHANGES;
                    }
                 else if ((pointer = strstr(buffer, "\n", buffer, base);
                    {
                       result        = CHANGES;
                    }
                 else
                    {
                       if (fputs(buffer, stdout) == 0)
                          {
```

```
                        cerr << "Could not write to stdout.\n";
                        result = ERROR;
                        break;
                     }
               }
         }
      return(result);
   }

int change_ram(const char* base)
   {
      int               result          = NOCHANGES;
      char              buffer[1024];
      char              newram[1024];

      if (fgets(buffer, 1024, stdin) == (char*)0)
         {
            /* Could not read from file.  */

            cerr << "Could not read from stdin.\n";
            result      = ERROR;
         }
      else
         {
            strcat(newram, base);
            strcat(newram, buffer + strlen(website) + 6);
            if (fputs(newram, stdout) == 0)
               {
                  /* Could not write to stdout.  */

                  cerr << "Could not write to stdout.\n";
                  result        = ERROR;
               }
            else
               {
                  result        = CHANGES;
               }
         }
      return(result);
   }

int main(int argc, char** argv)
   {
      int               result          = NOCHANGES;
      char              base[1024];

      if (parse_command_line(argc, argv) == ERROR)
         {
            cerr << "Error parsing shell program command line.\n";
```

```
      result      = ERROR;
   }
else if (strcmp(account, "padre") == 0)
   {
      strcpy(base, "http://padresoftware.com/");
   }
else if (strcmp(account, "tripod") == 0)
   {
      strcpy(base, "http://weedguy20.tripod.com/");
   }
else if (strcmp(account, "weedguy") == 0)
   {
      strcpy(base, "http://weedguy.padresoftware.com/");
   }
else if (strcmp(account, "president") == 0)
   {
      strcpy(base, "http://padresi.tripod.com/");
   }
else if (strcmp(account, "personal") == 0)
   {
      strcpy(base, "http://charles.padresoftware.com/");
   }
else if (strcmp(account, "journal") == 0)
   {
      strcpy(base, "http://videojournal.tripod.com/");
   }
else
   {
      cerr << "Unknown account!\n";
      result      = ERROR;
   }
if (result != ERROR)
   {
      if (strcmp(extension, "html") == 0)
         {
            result        = change_html(base);
         }
      else if (strcmp(extension, "ram") == 0)
         {
            result        = change_ram(base);
         }
      else
         {
            cerr << "Unknown extension!\n";
            result        = ERROR;
         }
   }
return(result);
```

```
}
```