HTMLDOC 1.8.20 Software Users Manual

ESP−003−20020507

Easy Software Products

Table of Contents

Table of Contents

Table of Contents

Introduction

This document describes how to use the *HTMLDOC* software, version 1.8.20. *HTMLDOC* converts Hyper−Text Markup Language ("HTML") input files into indexed HTML, Adobe® PostScript®, or Adobe Portable Document Format ("PDF") files.

HTMLDOC supports most HTML 3.2 elements, some HTML 4.0 elements, and can generate title and table of contents pages. It does not currently support stylesheets.

HTMLDOC can be used as a standalone application, in a batch document processing environment, or as a web−based report generation application.

No restrictions are placed upon the output produced by *HTMLDOC*.

History

Like many programs *HTMLDOC* was developed in response to a need our company had for generating high−quality documentation in printed and electronic forms. For a while we used FrameMaker® and a package from `sgi` that generated "compiled" Standard Generalized Markup Language ("SGML") files that could be used by the Electronic Book Technologies ("EBT") documentation products (EBT is now owned by INSO.) When `sgi` stopped supporting these tools we turned to INSO, but the cost of their tools is prohibitive to small businesses.

In the end we decided to write our own program to generate our documentation. HTML seemed to be the source format of choice since WYSIWYG HTML editors are widely (and freely) available and at worst you can use a plain text editor. We needed HTML output for documentation on our web server, PDF for customers

to read and/or print from their computers, and PostScript for our own printing needs.

The result of our efforts is the *HTMLDOC* software which is available for UNIX® and Microsoft® Windows®. Among other things, this software users manual is produced using *HTMLDOC*.

Organization of This Manual

This manual is organized into tutorial and reference chapters:

Support

Commercial support is available from Easy Software Products. Information can be found at the *HTMLDOC* web page, "http://www.easysw.com/htmldoc".

Encryption Support

HTMLDOC includes code to encrypt PDF document files using the RC4 algorithm with up to a 128−bit key. While this software and code may be freely used and exported under current US laws, other countries may restrict your use and possession of this code and software.

Copyright, Trademark, and License Information

This software is based in part on the work of the Independent JPEG Group and FLTK project.

Chapter 1 − Installing HTMLDOC

This chapter describes the steps needed to install *HTMLDOC* on your system from any of the source or binary distributions.

Installing a Binary Distribution

HTMLDOC binary distributions are available for a number of UNIX and Windows platforms.

Requirements

HTMLDOC requires approximately 2MB of disk space and one of the following environments:

- AIX 4.3 or higher
- Compaq Tru64 UNIX 4.0 or higher
- FreeBSD 4.5 or higher
- HP−UX 10.20 or higher
- IRIX 5.3 or higher
- Linux 2.0 or higher
- MacOS X 10.1 or higher
- Microsoft Windows 95/98/Me/NT 4.0/2000/XP
- Solaris 2.5 or higher

HTMLDOC may compile and run on other platforms, however we have not tested nor do we provide support for platforms other than those listed previously.

Installing HTMLDOC under Debian GNU/Linux

Run the following command to install *HTMLDOC* under Debian GNU/Linux:

```
% dselect install htmldoc-version-linux-2.0-intel.deb ENTER
```

Uninstalling HTMLDOC under Debian GNU/Linux

Run the following command to remove *HTMLDOC* under Debian GNU/Linux:

```
% dselect remove htmldoc ENTER
```

Installing HTMLDOC under Red Hat Linux

Run the following command to install *HTMLDOC* under Red Hat Linux:

```
% rpm -i htmldoc-version-linux-2.0-intel.rpm ENTER
```

Uninstalling HTMLDOC under Red Hat Linux

Run the following command to remove *HTMLDOC* from your Red Hat Linux system:

```
% rpm -e htmldoc ENTER
```

Installing HTMLDOC under UNIX

Run the following commands to install *HTMLDOC* under UNIX:

```
% gunzip htmldoc-version-platform.tar.gz ENTER
% tar xf htmldoc-version-platform.tar ENTER
% ./setup ENTER
```

Substitute the correct version and platform strings as appropriate.

Uninstalling HTMLDOC under UNIX

Run the following command to remove *HTMLDOC* from your UNIX system:

```
% /etc/software/htmldoc.remove ENTER
```

Installing HTMLDOC under MacOS X

Run the following commands to install *HTMLDOC* under MacOS X:

```
% tar xzf htmldoc-version-darwin-5.3-powerpc.tar.gz ENTER
% sudo ./htmldoc.install ENTER
```

Substitute the correct version and platform strings as appropriate.

Uninstalling HTMLDOC under MacOS X

Run the following command to remove *HTMLDOC* from your MacOS X system:

```
% sudo /etc/software/htmldoc.remove ENTER
```

Installing HTMLDOC under Windows

HTMLDOC is provided in a self−extracting installation file under Windows. Double−click on the setup icon to install *HTMLDOC* under Windows.

Uninstalling HTMLDOC under Windows

Open the Control Panel window and double−click on the *Add/Remove Software* icon. When the available software list is displayed, select *HTMLDOC* and click on the *Remove* button.

Installing HTMLDOC from the Source Distribution

The complete source to *HTMLDOC* is available to build *HTMLDOC* for different directories, architectures, or operating systems.

Requirements

HTMLDOC requires ANSI C and C++ compilers − recent versions of GCC/EGCS work fine. To build the GUI you'll also need:

- Fast Light Tool Kit ("FLTK"), version 1.0 or newer (version 1.1.x preferred).
- X11 libraries, R5 or higher (needed to build under UNIX and OS/2 only.)

Secure (https) URL support can be enabled via the OpenSSL library. You should use at least version 0.9.6.

Configuring the UNIX Source

HTMLDOC uses a configuration script produced by GNU autoconf to configure itself for your system. If your ANSI C compiler is not called *cc* or *gcc*, set the CC environment variable to the name and path of your ANSI C compiler:

```
% setenv CC /path/to/compiler ENTER          [C Shell]
% CC=/path/to/compiler; export CC ENTER      [Bourne/Korn Shell]
```

Similarly, if your C++ compiler is not called *CC*, *gcc*, *c++*, or *g++*, set the CXX environment variable to the name and path of your C++ compiler:

```
% setenv CXX /path/to/compiler ENTER         [C Shell]
% CXX=/path/to/compiler; export CXX ENTER    [Bourne/Korn Shell]
```

Then run the following command to configure *HTMLDOC* for installation in the default directories:

```
% ./configure ENTER
```

The default configuration will install *HTMLDOC* in the */usr/bin* directory with the data files under */usr/share/htmldoc* and the documentation and on−line help under */usr/share/doc/htmldoc*. Use the `--prefix` option to change the installation prefix to */usr/local*:

```
% ./configure --prefix=/usr/local ENTER
```

If the FLTK library is not installed in a standard location for your compilers, use the `--with-fltk-includes` and `--with-fltk-libs` options to point to the FLTK library:

```
% ./configure --with-fltk-libs=/path/to/fltk/lib \
        --with-fltk-includes=/path/to/fltk ENTER
```

Finally, if the OpenSSL library is not installed in a standard location for your compilers, use the `--with-openssl-includes` and `--with-openssl-libs` options to point to the OpenSSL library:

```
% ./configure --with-openssl-libs=/path/to/openssl/lib \
        --with-openssl-includes=/path/to/openssl ENTER
```

Compiling under UNIX

HTMLDOC is built from a Makefile in the distribution's main directory. Simply run the "make" command to build *HTMLDOC*:

```
% make ENTER
```

If you get any fatal errors, please subscribe to the *HTMLDOC* mailing list and send a copy of the make/compiler output to "htmldoc@easysw.com" for assistance. Please note the version of *HTMLDOC* that you are using as well as any pertinent system information (operating system, OS version, compiler, etc.)

To subscribe to the *HTMLDOC* mailing list, send a message to "majordomo@easysw.com" with the text:

```
subscribe htmldoc
```

in the message body. *You must subscribe to the list to post questions and comments.*

Installing under UNIX

To install *HTMLDOC* simply run the "make install" command:

```
% make install ENTER
```

If you are installing in a restricted directory like */usr* then you'll need to be logged in as root.

Compiling with Visual C++

A Visual C++ 6.0 workspace file and associated project files are included in the source distribution under the "visualc" directory. Open the workspace file "htmldoc.dsw", adjust the FLTK include and project file locations, and then build the *HTMLDOC* target.

Installing with Visual C++

The Windows installation package is created using InstallShield for Visual C++ 6. The "visualc/HTMLDOC" directory contains the installation information for *HTMLDOC* needed to build a binary distribution with InstallShield.

To install *HTMLDOC* without InstallShield, create an installation directory and copy the *htmldoc.exe* executable, the *afm* directory, the *data* directory, and the *doc* directory to it.

Then use the *regedit* program to create the following two string entries:

HKEY_LOCAL_MACHINE\Software\Easy Software Products\HTMLDOC\data
: C:\installation\directory

HKEY_LOCAL_MACHINE\Software\Easy Software Products\HTMLDOC\doc
: C:\installation\directory\doc

Chapter 2 − Getting Started

This chapter describes how to start *HTMLDOC* and convert HTML files into PostScript and PDF files.

> **Note:**
>
> *HTMLDOC* currently does not support HTML 4.0 features such as stylesheets or the `STYLE`, `TBODY`, `THEAD`, or `TFOOT` elements. For more information, please consult Chapter 6 − HTML Reference.

Starting HTMLDOC

To start *HTMLDOC* under UNIX type:

```
% htmldoc ENTER
```

Choose *HTMLDOC* from the *Start* menu to start *HTMLDOC* under Windows.

Choosing a HTML File

The *HTMLDOC* window (Figure 2−1) shows the list of input files that will be converted. Start by clicking on the *Web Page* radio button (1) to specify that you will be converting a HTML web page file.

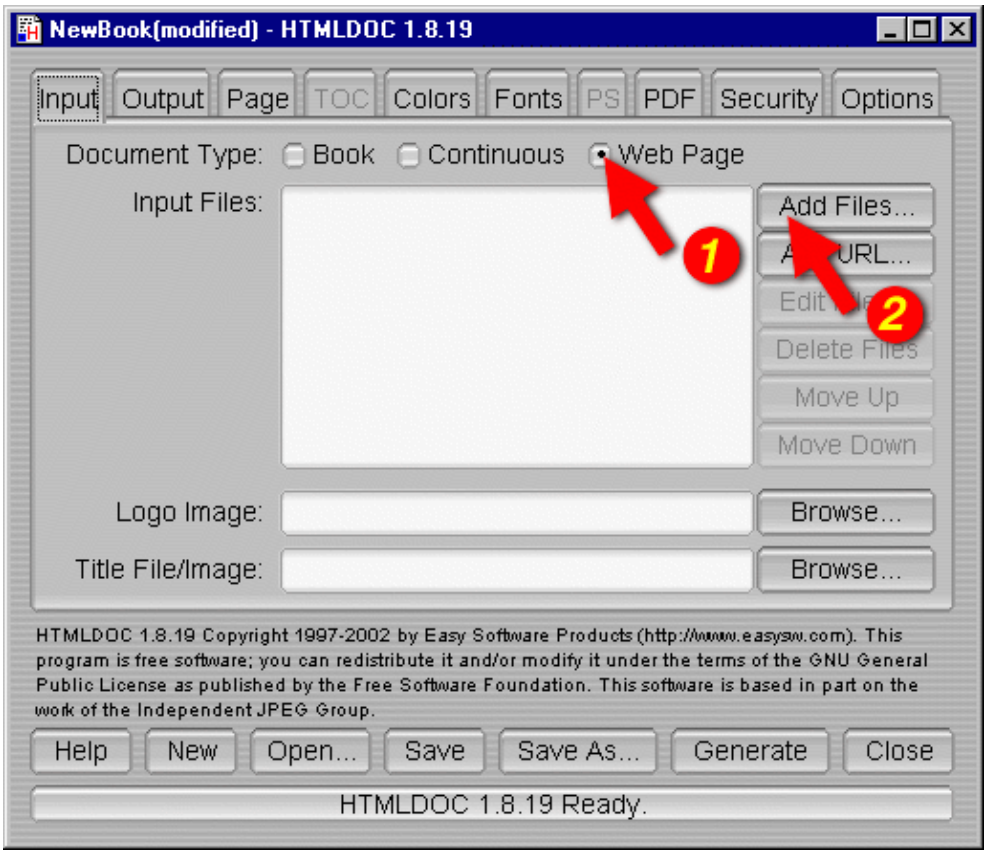

Figure 2−1 − The HTMLDOC Window

Then choose a file for conversion by clicking on the *Add Files...* button (2). When the file chooser dialog appears (Figure 2−2), double−click on the HTML file (3) you wish to convert from the list of files.

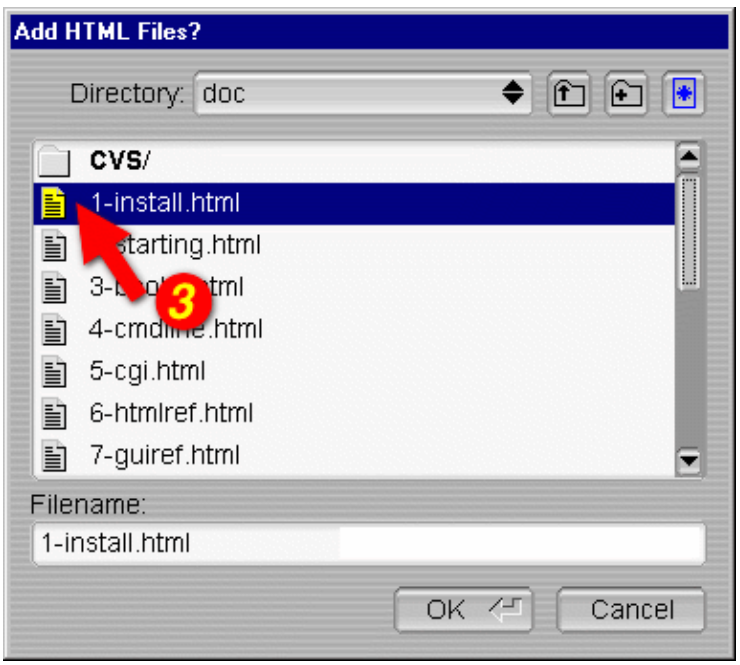

Figure 2−2 − The File Chooser Dialog

Setting the Output File

Now that you've chosen a HTML file to convert, click on the *Output* tab (4) to set the output file (Figure 2−3). Type the name of the output file into the *Output Path* field or click on the *Browse...* button (5) to select the output file using the file chooser.

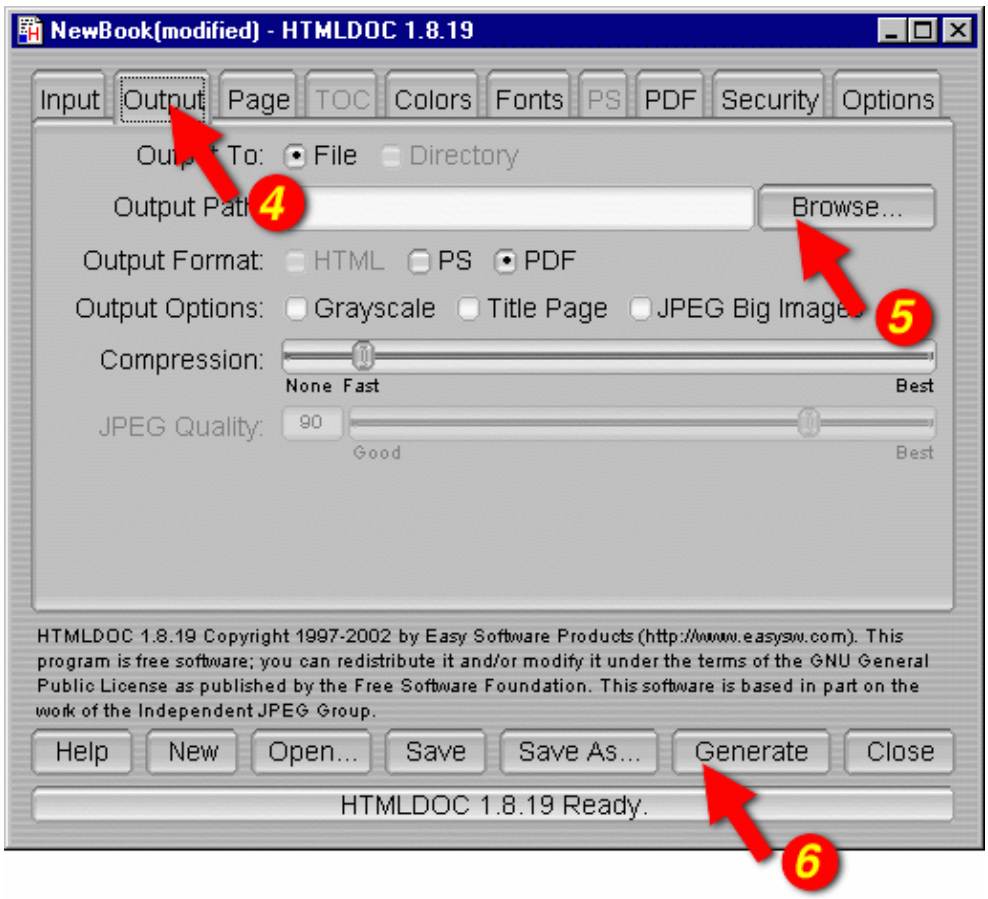

Figure 2−3 − The Output Tab

Since you chose to convert a *Web Page* instead of a book, *HTMLDOC* has automatically chosen to produce a PDF file.

Generating the Document

Once you have chosen the output file you can generate it by clicking on the *Generate* button (6) at the bottom of the *HTMLDOC* window. When the conversion is completed you can open the PDF file that is produced using Adobe Acrobat Reader or any other PDF viewing application.

Chapter 3 − Generating Books

This chapter describes how to generate whole books from HTML files.

Overview

While *HTMLDOC* can convert web pages into PostScript and PDF files, its real strength is generating indexed HTML, PostScript, or PDF books.

HTMLDOC uses HTML heading elements to delineate chapters and headings in a book. The `H1` element is used for chapters:

```
<HTML>
<HEAD>
    <TITLE>The Little Computer that Could</TITLE>
</HEAD>
<BODY>
<H1>Chapter 1 − The Little Computer is Born</H1>
...
<H1>Chapter 2 − Little Computer's First Task</H1>
...
</BODY>
</HTML>
```

Sub−headings are marked using the `H2` through `H6` elements.

Note:

When using book mode, HTMLDOC starts rendering with the first `H1` element. Any text, images, tables, and other viewable elements that precede the first `H1` element are silently ignored.

Choosing HTML Files

Start by clicking on the *Book* radio button (1) to specify you'll be converting one or more HTML files into a book.

Then choose one or more files for conversion by clicking on the *Add Files...* button (2). When the file chooser dialog appears, pick the file(s) you wish to convert from the list of files and then click on the *OK* button.

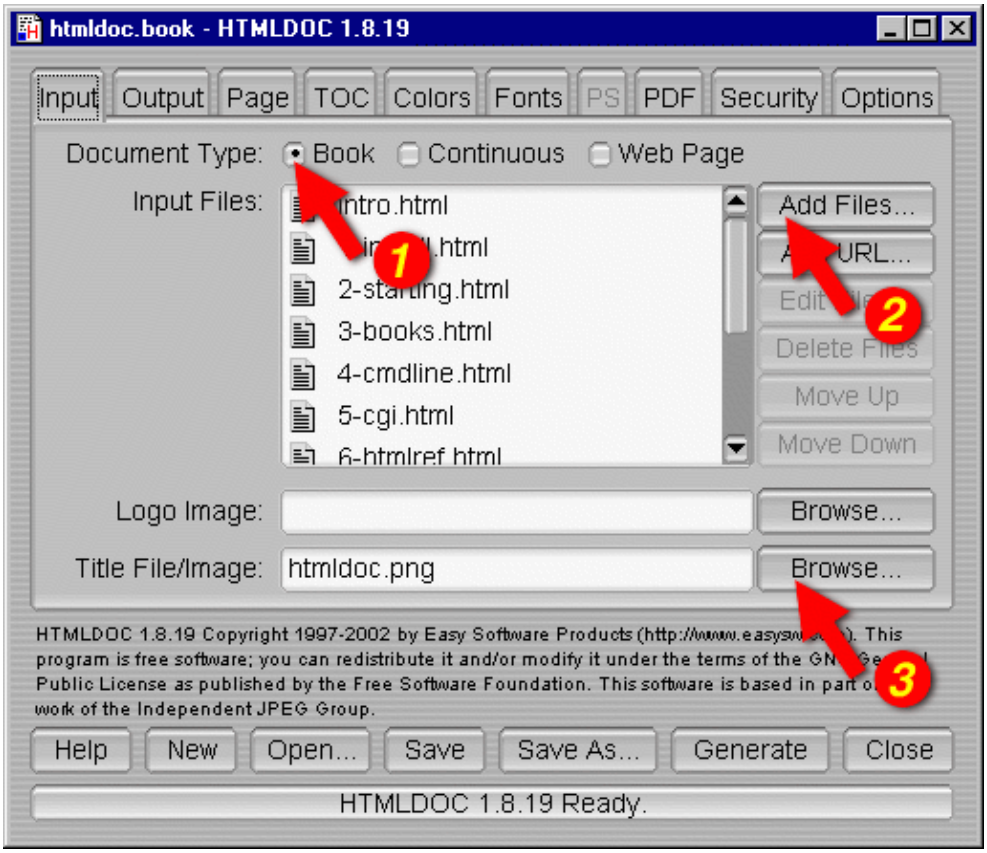

Figure 3−1: The Input Tab

Selecting a Title File

HTMLDOC supports automatic generation of a title page using an image file, the title text, and other `META` information on it. Type the title image filename into the *Title File* field or click on the *Browse...* button (3) to select a title image for your book. *HTMLDOC* can also use a HTML file that you have generated for the title page(s). To use a HTML title page, type the title filename into the *Title File* field or click on the *Browse...* button (3) to select a HTML file for your book.

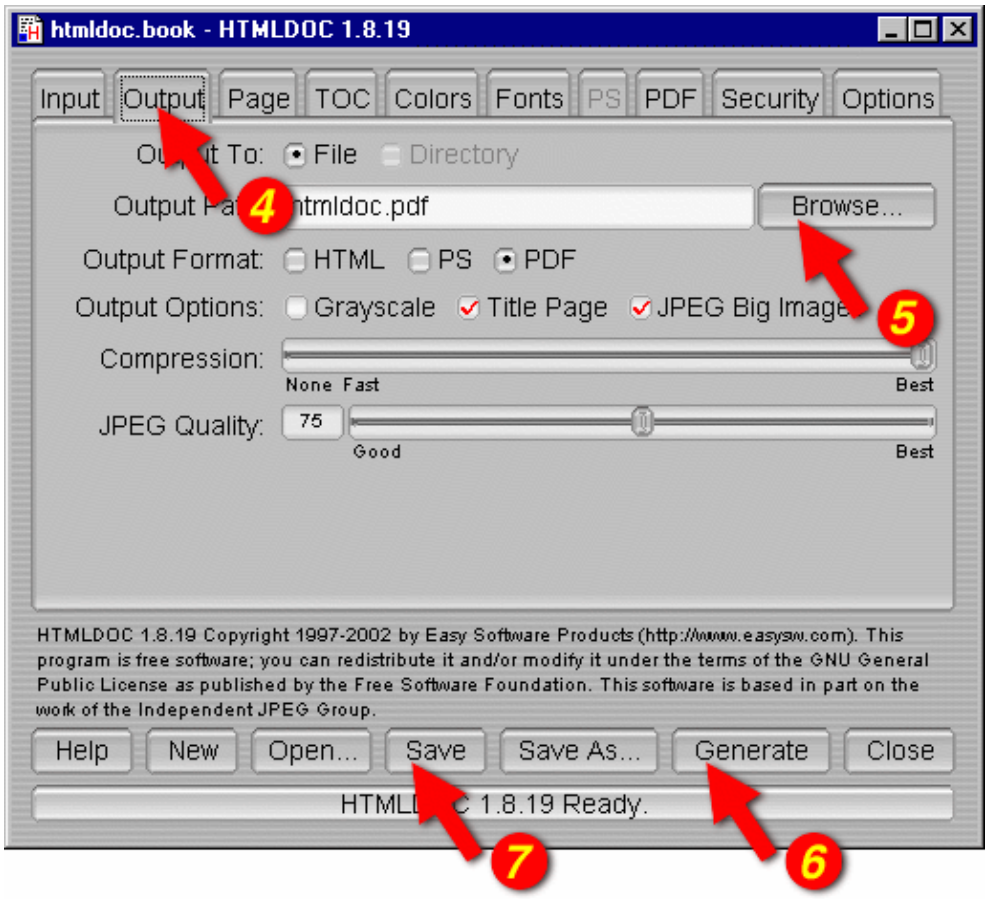

Figure 3−2: The Output Tab

Setting the Output Format

The output format is set in the *Output* tab (4). Click on the *Output* tab and then click on the *HTML*, *PS*, or *PDF* radio buttons to set the output format.

Setting the Output File

Now that you've chosen an output format, type the name of the output file into the *Output Path* field or click on the *Browse...* button (5) to select the output file using the file chooser.

Generating the Document

Once you have chosen the output file you can generate it by clicking on the *Generate* button (6) at the bottom of the *HTMLDOC* window.

Saving Your Book

HTMLDOC can save the list of HTML files, the title file, and all other options to a special `.BOOK` file so you can regenerate your book when you make changes to your HTML files.

Click on the *Save* button (7) to save the current book to a file.

Chapter 4 − HTMLDOC from the Command−Line

This chapter describes how to use *HTMLDOC* from the command−line to convert web pages and generate books.

> **Note:**
>
> The free version of *HTMLDOC* for Windows does not include the command−line program.

Converting Web Pages

To convert a single web page type:

```
% htmldoc --webpage -f output.pdf filename.html ENTER
% htmldoc --webpage -f output.ps filename.html ENTER
```

To convert more than one web page with page breaks between each HTML file, type:

```
% htmldoc --webpage -f output.pdf file1.html ... fileN.html ENTER
% htmldoc --webpage -f output.ps file1.html ... fileN.html ENTER
```

The `--webpage` option tells *HTMLDOC* that you want to convert web pages or other unstructured HTML files. You can also use `--continuous` to convert multiple HTML files without page breaks between files and `--book` to convert structured HTML files with headings into a book with a table of contents. The default document type is `--book`.

The `-f` option tells *HTMLDOC* the file to generate. If you don't specify an output file, a PDF file is sent to the standard output. The `output.pdf` and `output.ps` arguments are the names of the output files you want to generate. The `.pdf` extension specifies that you want to generate a PDF file, while the `.ps` extension specifies PostScript output.

The `filename.html`, `file1.html`, and `fileN.html` arguments are the input HTML files you want to convert. The HTML files can also be URLs, for example:

```
% htmldoc --webpage -f output.pdf http://slashdot.org/ ENTER
% htmldoc --webpage -f output.ps http://freshmeat.net/ http://www.easysw.com/ ENTER
```

Generating Books

Type one of the following commands to generate a book from one or more HTML files:

```
% htmldoc --book -f output.html file1.html ... fileN.html ENTER
% htmldoc --book -f output.pdf file1.html ... fileN.html ENTER
% htmldoc --book -f output.ps file1.html ... fileN.html ENTER
```

where `output.html`, `output.pdf`, and `output.ps` are the names of the files you want to generate, and `file1.html` to `fileN.html` are the HTML files you want to use for the book.

The `--book` option tells *HTMLDOC* that you want to generate a book from the HTML file(s) you specified.

The `-f` option tells *HTMLDOC* what file to generate. If you don't specify an output file then a PDF file is sent to the standard output.

HTMLDOC will build a table of contents for the book using the heading elements (`H1`, `H2`, etc.) in your HTML files. It will also add a title page using the document `TITLE` text and other `META` information you supply in your HTML files. See Chapter 6 − HTML Reference for more information on the `META` variables that are supported.

> **Note:**
>
> When using book mode, HTMLDOC starts rendering with the first `H1` element. Any text, images, tables, and other viewable elements that precede the first `H1` element are silently ignored.

Setting the Title File

The `--titlefile` option sets the HTML file or image to use on the title page:

```
% htmldoc --titlefile filename.bmp ... ENTER
% htmldoc --titlefile filename.gif ... ENTER
% htmldoc --titlefile filename.jpg ... ENTER
% htmldoc --titlefile filename.png ... ENTER
% htmldoc --titlefile filename.html ... ENTER
```

HTMLDOC supports BMP, GIF, JPEG, and PNG images, as well as generic HTML text you supply for the title page(s).

Chapter 5 − Using HTMLDOC on a Web Server

This chapter describes how to interface *HTMLDOC* to your web server using CGI scripts and programs.

> **Note:**
>
> The free version of *HTMLDOC* for Windows does not support use from a web server.

The Basics

HTMLDOC can be used in a variety of ways to generate formatted reports on a web server. The most common way is to combine *HTMLDOC* with a CGI script or program and send the output to the HTTP client.

To make this work the CGI script or program must send the appropriate HTTP attributes, the required empty line to signify the beginning of the document, and then execute the *HTMLDOC* program to generate the HTML, PostScript, or PDF file as needed.

Another way to generate PDF files from your reports is to use *HTMLDOC* as a "portal" application. When used as a portal, *HTMLDOC* automatically retrieves the named document or report from your server and passes a PDF version to the web browser. See the next sections for more information.

WARNING:

Passing information directly from the web browser to *HTMLDOC* can potentially expose your system to security risks. Always be sure to "sanitize" any input from the web browser so that filenames, URLs, and options passed to *HTMLDOC* are not acted on by the shell program.

Calling HTMLDOC from a Shell Script

Shell scripts are probably the easiest to work with, but are normally limited to GET type requests. Here is a script called *topdf* that acts as a portal, converting the named file to PDF:

```
#!/bin/sh
#
# Sample "portal" script to convert the named HTML file to PDF on-the-fly.
#
# Usage: http://www.domain.com/path/topdf/path/filename.html
#

#
# The "options" variable contains any options you want to pass to HTMLDOC.
#

options="-t pdf --webpage --header ... --footer ..."

#
# Tell the browser to expect a PDF file...
#

echo "Content-Type: application/pdf"
echo ""

#
# Run HTMLDOC to generate the PDF file...
#

htmldoc $options http://${SERVER_NAME}:${SERVER_PORT}$PATH_INFO
```

Users of this CGI would reference the URL "http://www.domain.com/topdf.cgi/index.html" to generate a PDF file of the site's home page.

The *options* variable in the script can be set to use any supported command-line option for *HTMLDOC*; for a complete list see Chapter 8 - Command-Line Reference.

Calling HTMLDOC from Perl

Perl scripts offer the ability to generate more complex reports, pull data from databases, etc. The easiest way to interface Perl scripts with *HTMLDOC* is to write a report to a temporary file and then execute *HTMLDOC* to generate the PDF file.

Here is a simple Perl subroutine that can be used to write a PDF report to the HTTP client:

```
sub topdf(filename);

sub topdf {
    # Get the filename argument...
    my $filename = shift;

    # Make stdout unbuffered...
    select(STDOUT); $| = 1;

    # Write the content type to the client...
    print "Content-Type: application/pdf\n\n";

    # Run HTMLDOC to provide the PDF file to the user...
    system "htmldoc -t pdf --quiet --webpage $filename";
}
```

Calling HTMLDOC from PHP

PHP is quickly becoming the most popular server-side scripting language available. PHP provides a `passthru()` function that can be used to run *HTMLDOC*. This combined with the `header()` function can be used to provide on-the-fly reports in PDF format.

Here is a simple PHP function that can be used to convert a HTML report to PDF and send it to the HTTP client:

```
function topdf($filename, $options = "") {
    # Write the content type to the client...
    header("Content-Type: application/pdf");
    flush();

    # Run HTMLDOC to provide the PDF file to the user...
    passthru("htmldoc -t pdf --quiet --jpeg --webpage $options '$filename'");
}
```

The function accepts a filename and an optional "options" string for specifying the header, footer, fonts, etc.

To prevent malicious users from passing in unauthorized characters into this function, the following function can be used to verify that the URL/filename does not contain any characters that might be interpreted by the shell:

```
function bad_url($url) {
    // See if the URL starts with http: or https:...
    if (strncmp($url, "http://", 7) != 0 &&
        strncmp($url, "https://", 8) != 0) {
        return 1;
    }

    // Check for bad characters in the URL...
    $len = strlen($url);
    for ($i = 0; $i < $len; $i ++) {
        if (!strchr("~_*()/:%?+-&@;=,$.", $url[$i]) &&
            !ctype_alnum($url[$i])) {
            return 1;
        }
    }

    return 0;
}
```

Another method is to use the `escapeshellarg()` function provided with PHP 4.0.3 and higher to generate a quoted shell argument for *HTMLDOC*.

To make a "portal" script, add the following code to complete the example:

```
global $SERVER_NAME;
global $SERVER_PORT;
global $PATH_INFO;
global $QUERY_STRING;

if ($QUERY_STRING != "") {
    $url = "http://${SERVER_NAME}:${SERVER_PORT}${PATH_INFO}?${QUERY_STRING}";
} else {
    $url = "http://${SERVER_NAME}:${SERVER_PORT}$PATH_INFO";
}

if (bad_url($url)) {
  print("<HTML><HEAD><TITLE>Bad URL</TITLE></HEAD>\n"
       ."<BODY><H1>Bad URL</H1>\n",
       ."<P>The URL <B><TT>$url</TT></B> is bad.</P>\n"
       ."</BODY></HTML>\n");
} else {
  topdf($url);
}
```

Calling HTMLDOC from C

C programs offer the best flexibility and easily supports on−the−fly report generation without the need for temporary files.

Here are some simple C functions that can be used to generate a PDF report to the HTTP client from a temporary file or pipe:

```
#include <stdio.h>
#include <stdlib.h>


/* topdf() − convert a HTML file to PDF */
FILE *topdf(const char *filename)       /* HTML file to convert */
{
  char  command[1024];                  /* Command to execute */


  puts("Content−Type: application/pdf\n");

  sprintf(command, "htmldoc −t pdf −−webpage %s", filename);

  return (popen(command, "w"));
}


/* topdf2() − pipe HTML output to HTMLDOC for conversion to PDF */
FILE *topdf2(void)
{
  puts("Content−Type: application/pdf\n");
  return (popen("htmldoc −t pdf −−webpage −", "w"));
}
```

Calling HTMLDOC from Java

Java programs are a portable way to add PDF support to your web server. Here is a class called *htmldoc* that acts as a portal, converting the named file to PDF. It can also be called by your Java servlets to process an HTML file and send the result to the client in PDF format:

```
class htmldoc
{
  // Convert named file to PDF on stdout...
  public static int topdf(String filename)// I − Name of file to convert
  {
    String              command;        // Command string
    Process             process;        // Process for HTMLDOC
    Runtime             runtime;        // Local runtime object
    java.io.InputStream input;          // Output from HTMLDOC
    byte                buffer [];      // Buffer for output data
    int                 bytes;          // Number of bytes


    // First tell the client that we will be sending PDF...
    System.out.print("Content−type: application/pdf\n\n");

    // Construct the command string
    command = "htmldoc −−quiet −−jpeg −−webpage −t pdf −−left 36 " +
              "−−header .t. −−footer .1. " + filename;

    // Run the process and wait for it to complete...
    runtime = Runtime.getRuntime();

    try
    {
      // Create a new HTMLDOC process...
      process = runtime.exec(command);

      // Get stdout from the process and a buffer for the data...
      input  = process.getInputStream();
      buffer = new byte[8192];

      // Read output from HTMLDOC until we have it all...
      while ((bytes = input.read(buffer)) > 0)
        System.out.write(buffer, 0, bytes);

      // Return the exit status from HTMLDOC...
      return (process.waitFor());
    }
    catch (Exception e)
    {
      // An error occurred − send it to stderr for the web server...
      System.err.print(e.toString() + " caught while running:\n\n");
      System.err.print("    " + command + "\n");
      return (1);
    }
  }

  // Main entry for htmldoc class
  public static void main(String[] args)// I − Command−line args
  {
    String      server_name,            // SERVER_NAME env var
                server_port,            // SERVER_PORT env var
```

```
                path_info,              // PATH_INFO env var
                query_string,           // QUERY_STRING env var
                filename;               // File to convert


    if ((server_name = System.getProperty("SERVER_NAME")) != null &&
        (server_port = System.getProperty("SERVER_PORT")) != null &&
        (path_info = System.getProperty("PATH_INFO")) != null)
    {
      // Construct a URL for the resource specified...
      filename = "http://" + server_name + ":" + server_port + path_info;

      if ((query_string = System.getProperty("QUERY_STRING")) != null)
      {
        filename = filename + "?" + query_string;
      }
    }
    else if (args.length == 1)
    {
      // Pull the filename from the command-line...
      filename = args[0];
    }
    else
    {
      // Error - no args or env variables!
      System.err.print("Usage: htmldoc.class filename\n");
      return;
    }

    // Convert the file to PDF and send to the web client...
    topdf(filename);
  }
}
```

Chapter 6 − HTML Reference

This chapter defines all of the HTML elements and attributes that are recognized and supported by *HTMLDOC*.

General Usage

There are two types of HTML files − structured documents using headings (H1, H2, etc.) which *HTMLDOC* calls "books", and unstructured documents that do not use headings which *HTMLDOC* calls "web pages".

A very common mistake is to try converting a web page using:

```
htmldoc -f filename.pdf filename.html
```

which will likely produce a PDF file with no pages. To convert web page files you **must** use the `--webpage` option at the command−line or choose *Web Page* in the input tab of the GUI.

***HTMLDOC* does not support HTML 4.0 elements, attributes, stylesheets, or scripting.**

Elements

The following HTML elements are recognized by *HTMLDOC*:

Element	Version	Supported?	Notes
!DOCTYPE	3.0	Yes	DTD is ignored
A	1.0	Yes	See Below
ACRONYM	2.0	Yes	No font change
ADDRESS	2.0	Yes	
AREA	2.0	No	
B	1.0	Yes	
BASE	2.0	No	
BASEFONT	1.0	No	
BIG	2.0	Yes	
BLINK	2.0	No	
BLOCKQUOTE	2.0	Yes	
BODY	1.0	Yes	
BR	2.0	Yes	
CAPTION	2.0	Yes	See Below
CENTER	2.0	Yes	
CITE	2.0	Yes	Italic/Oblique
CODE	2.0	Yes	Courier
DD	2.0	Yes	
DEL	2.0	Yes	Strikethrough
DFN	2.0	Yes	Helvetica
DIR	2.0	Yes	
DIV	3.2	Yes	
DL	2.0	Yes	
DT	2.0	Yes	Italic/Oblique
EM	2.0	Yes	Italic/Oblique
EMBED	2.0	Yes	HTML Only
FONT	2.0	Yes	See Below

Element	Version	Supported?	Notes
FORM	2.0	No	
FRAME	3.2	No	
FRAMESET	3.2	No	
H1	1.0	Yes	Boldface, See Below
H2	1.0	Yes	Boldface, See Below
H3	1.0	Yes	Boldface, See Below
H4	1.0	Yes	Boldface, See Below
H5	1.0	Yes	Boldface, See Below
H6	1.0	Yes	Boldface, See Below
HEAD	1.0	Yes	
HR	1.0	Yes	See Below
HTML	1.0	Yes	
I	1.0	Yes	
IMG	1.0	Yes	See Below
INPUT	2.0	No	
INS	2.0	Yes	Underline
ISINDEX	2.0	No	
KBD	2.0	Yes	Courier Bold
LI	2.0	Yes	
LINK	2.0	No	
MAP	2.0	No	
MENU	2.0	Yes	
META	2.0	Yes	See Below
MULTICOL	N3.0	No	
NOBR	1.0	No	
NOFRAMES	3.2	No	
OL	2.0	Yes	
OPTION	2.0	No	
P	1.0	Yes	
PRE	1.0	Yes	

Element	Version	Supported?	Notes
S	2.0	Yes	Strikethrough
SAMP	2.0	Yes	Courier
SCRIPT	2.0	No	
SELECT	2.0	No	
SMALL	2.0	Yes	
SPACER	N3.0	Yes	
STRIKE	2.0	Yes	
STRONG	2.0	Yes	Boldface Italic/Oblique
SUB	2.0	Yes	Reduced Fontsize
SUP	2.0	Yes	Reduced Fontsize
TABLE	2.0	Yes	See Below
TD	2.0	Yes	
TEXTAREA	2.0	No	
TH	2.0	Yes	Boldface Center
TITLE	2.0	Yes	
TR	2.0	Yes	
TT	2.0	Yes	Courier
U	1.0	Yes	
UL	2.0	Yes	
VAR	2.0	Yes	Helvetica Oblique
WBR	1.0	No	

Comments

HTMLDOC supports many special HTML comments to initiate page breaks, set the header and footer text, and control the current media options:

`<!-- FOOTER LEFT "foo" -->`
: Sets the left footer text; the test is applied to the current page if empty, or the next page otherwise.

`<!-- FOOTER CENTER "foo" -->`
: Sets the center footer text; the test is applied to the current page if empty, or the next page otherwise.

`<!-- FOOTER RIGHT "foo" -->`
Sets the right footer text; the test is applied to the current page if empty, or the next page otherwise.

`<!-- HALF PAGE -->`
Break to the next half page.

`<!-- HEADER LEFT "foo" -->`
Sets the left header text; the test is applied to the current page if empty, or the next page otherwise.

`<!-- HEADER CENTER "foo" -->`
Sets the center header text; the test is applied to the current page if empty, or the next page otherwise.

`<!-- HEADER RIGHT "foo" -->`
Sets the right header text; the test is applied to the current page if empty, or the next page otherwise.

`<!-- MEDIA BOTTOM nnn -->`
Sets the bottom margin of the page. The "nnn" string can be any standard measurement value, e.g. 0.5in, 36, 12mm, etc. Breaks to a new page if the current page is already marked.

`<!-- MEDIA COLOR "foo" -->`
Sets the media color attribute for the page. The "foo" string is any color name that is supported by the printer, e.g. "Blue", "White", etc. Breaks to a new page or sheet if the current page is already marked.

`<!-- MEDIA DUPLEX NO -->`
Chooses single−sided printing for the page; breaks to a new page or sheet if the current page is already marked.

`<!-- MEDIA DUPLEX YES -->`
Chooses double−sided printing for the page; breaks to a new sheet if the current page is already marked.

`<!-- MEDIA LANDSCAPE NO -->`
Chooses portrait orientation for the page; breaks to a new page if the current page is already marked.

`<!-- MEDIA LANDSCAPE YES -->`
Chooses landscape orientation for the page; breaks to a new page if the current page is already marked.

`<!-- MEDIA LEFT nnn -->`
Sets the left margin of the page. The "nnn" string can be any standard measurement value, e.g. 0.5in, 36, 12mm, etc. Breaks to a new page if the current page is already marked.

`<!-- MEDIA POSITION nnn -->`
Sets the media position attribute (input tray) for the page. The "nnn" string is an integer that usually specifies the tray number. Breaks to a new page or sheet if the current page is already marked.

`<!-- MEDIA RIGHT nnn -->`
Sets the right margin of the page. The "nnn" string can be any standard measurement value, e.g. 0.5in, 36, 12mm, etc. Breaks to a new page if the current page is already marked.

`<!-- MEDIA SIZE foo -->`
Sets the media size to the specified size. The "foo" string can be "Letter", "Legal", "Universal", or "A4" for standard sizes or "WIDTHxHEIGHTunits" for custom sizes, e.g. "8.5x11in"; breaks to a new page or sheet if the current page is already marked.

`<!-- MEDIA TOP nnn -->`
Sets the top margin of the page. The "nnn" string can be any standard measurement value, e.g. 0.5in, 36, 12mm, etc. Breaks to a new page if the current page is already marked.

`<!-- MEDIA TYPE "foo" -->`
Sets the media type attribute for the page. The "foo" string is any type name that is supported by the printer, e.g. "Plain", "Glossy", etc. Breaks to a new page or sheet if the current page is already marked.

<!-- NEED length -->
: Break if there is less than `length` units left on the current page. The `length` value defaults to lines of text but can be suffixed by `in`, `mm`, or `cm` to convert from the corresponding units.

<!-- NEW PAGE -->
: Break to the next page.

<!-- NEW SHEET -->
: Break to the next sheet.

<!-- NUMBER-UP nn -->
: Sets the number of pages that are placed on each output page. Valid values are 1, 2, 4, 6, 9, and 16.

<!-- PAGE BREAK -->
: Break to the next page.

Header/Footer Strings

The `HEADER` and `FOOTER` comments allow you to set an arbitrary string of text for the left, center, and right headers and footers. Each string consists of plain text; special values or strings can be inserted using the dollar sign ($):

$$
: Inserts a single dollar sign in the header.

CHAPTER
: Inserts the current chapter heading.

$CHAPTERPAGE
$CHAPTERPAGE(format)
: Inserts the current page number within a chapter or file. When a format is specified, uses that numeric format (1 = decimal, i = lowercase roman numerals, I = uppercase roman numerals, a = lowercase ascii, A = uppercase ascii) for the page numbers.

$CHAPTERPAGES
$CHAPTERPAGES(format)
: Inserts the total page count within a chapter or file. When a format is specified, uses that numeric format (1 = decimal, i = lowercase roman numerals, I = uppercase roman numerals, a = lowercase ascii, A = uppercase ascii) for the page count.

$DATE
: Inserts the current date.

$HEADING
: Inserts the current heading.

$LOGOIMAGE
: Inserts the logo image; all other text in the string will be ignored.

$PAGE
$PAGE(format)
: Inserts the current page number. When a format is specified, uses that numeric format (1 = decimal, i = lowercase roman numerals, I = uppercase roman numerals, a = lowercase ascii, A = uppercase ascii) for the page numbers.

$PAGES
$PAGES(format)
: Inserts the total page count. When a format is specified, uses that numeric format (1 = decimal, i = lowercase roman numerals, I = uppercase roman numerals, a = lowercase ascii, A = uppercase ascii) for the page count.

$TIME
: Inserts the current time.

$TITLE
: Inserts the document title.

FONT Attributes

Limited typeface specification is currently supported to ensure portability across platforms and for older PostScript printers:

Requested Font	Actual Font
Arial	Helvetica
Courier	Courier
Helvetica	Helvetica
Monospace	Courier
Sans−Serif	Helvetica
Serif	Times
Symbol	Symbol
Times	Times

All other unrecognized typefaces are silently ignored.

Headings

Currently *HTMLDOC* supports a maximum of 10000 headings and 100 chapters. These limits can be increased by changing the constants in the *config.h* file included with the source code.

All chapters start with a top−level heading (H1) markup. Any headings within a chapter must be of a lower level (H2 to H6). Each chapter starts a new page or the next odd−numbered page if duplexing is selected.

The headings you use within a chapter must start at level 2 (H2). If you skip levels the heading will be shown under the last level that was known. For example, if you use the following hierarchy of headings:

```
<H1>Chapter Heading</H1>
...
<H2>Section Heading 1</H2>
...
<H2>Section Heading 2</H2>
...
<H3>Sub−Section Heading 1</H3>
```

```
...
<H4>Sub−Sub−Section Heading 1</H4>
...
<H4>Sub−Sub−Section Heading 2</H4>
...
<H3>Sub−Section Heading 2</H3>
...
<H2>Section Heading 3</H2>
...
<H4>Sub−Sub−Section Heading 3</H4>
...
```

the table−of−contents that is generated will show:

Chapter Heading

- Section Heading 1
- Section Heading 2
  - Sub−Section Heading 1
    - Sub−Sub−Section Heading 1
    - Sub−Sub−Section Heading 2
  - Sub−Section Heading 2
    - Sub−Sub−Section Heading 3
- Section Heading 3

Numbered Headings

When the numbered headings option is enabled, *HTMLDOC* recognizes the following additional attributes for all heading elements:

VALUE="#"
: Specifies the starting value for this heading level (default is "1" for all new levels).

TYPE="1"
: Specifies that decimal numbers should be generated for this heading level.

TYPE="a"
: Specifies that lowercase letters should be generated for this heading level.

TYPE="A"
: Specifies that uppercase letters should be generated for this heading level.

TYPE="i"
: Specifies that lowercase roman numerals should be generated for this heading level.

TYPE="I"
: Specifies that uppercase roman numerals should be generated for this heading level.

Images

HTMLDOC supports loading of BMP, GIF, JPEG, and PNG image files. EPS and other types of image files are not supported at this time.

Links

External URL and internal (`#target` and `filename.html`) links are fully supported for HTML and PDF output.

When generating PDF files, local PDF file links will be converted to external file links for the PDF viewer instead of URL links. That is, you can directly link to another local PDF file from your HTML document with:

```
<A HREF="filename.pdf">...</A>
```

META Attributes

HTMLDOC supports the following `META` attributes for the title page and document information:

<META NAME="AUTHOR" CONTENT="...">
: Specifies the document author.

<META NAME="COPYRIGHT" CONTENT="...">
: Specifies the document copyright.

<META NAME="DOCNUMBER" CONTENT="...">
: Specifies the document number.

<META NAME="GENERATOR" CONTENT="...">
: Specifies the application that generated the HTML file.

<META NAME="KEYWORDS" CONTENT="...">
: Specifies document search keywords.

<META NAME="SUBJECT" CONTENT="...">
: Specifies document subject.

Page Breaks

HTMLDOC supports four new page comments to specify page breaks. In addition, the older `BREAK` attribute is still supported by the `HR` element:

```
<HR BREAK>
```

Support for the `BREAK` attribute is deprecated and will be removed in a future release of *HTMLDOC*.

Tables

Currently *HTMLDOC* supports a maximum of 200 columns within a single table. This limit can be increased by changing the `MAX_COLUMNS` constant in the *config.h* file included with the source code. *HTMLDOC* supports HTML 3.0 tables with the following exceptions:

- The `CAPTION` element is always shown at the top of the table.

***HTMLDOC* does not support HTML 4.0 table elements or attributes, such as `TBODY`, `THEAD`, `TFOOT`, or `RULES`.**

Chapter 7 − GUI Reference

This chapter describes all of the GUI controls in *HTMLDOC*.

The HTMLDOC GUI

The *HTMLDOC* GUI (Figures 7−1 through 7−11) is contained in a single window showing the input, output, and generation options. At the bottom are buttons to load, save, and generate documents.

Document File Operations

HTMLDOC stores the HTML files, settings, and options in `.BOOK` files. The buttons on the bottom of the *HTMLDOC* window allow you to manage these files and generate formatted documents.

New

The *New* button starts a new document. A confirmation dialog will appear if you have not saved the changes to the existing document.

Open...

The *Open...* button retrieves a document that you have saved previously. A file chooser dialog is displayed that allows you to pick an existing book file.

Save

The *Save* button saves the current document. A file chooser dialog is displayed if there is no filename assigned to the current document.

Note: Saving a document is not the same as *generating* a document. The book files saved to disk by the *Save* and *Save As...* buttons are ***not*** the final HTML, PDF, or PostScript output files. You generate those files by clicking on the *Generate* button.

Save As...

The *Save As...* button saves the current document to a new file. A file chooser dialog is displayed to allow you to specify the new document filename.

Note: Saving a document is not the same as *generating* a document. The book files saved to disk by the *Save* and *Save As...* buttons are ***not*** the final HTML, PDF, or PostScript output files. You generate those files by clicking on the *Generate* button.

Generate

The *Generate* button generates the current document, creating the specified HTML, PDF, or PostScript file(s) as needed. The progress meter at the bottom of the window will show the progress as each page or file is formatted and written.

Note: Generating a document is not the same as ***saving*** a document. To save the current HTML files and settings in the *HTMLDOC* GUI, click on the *Save* or *Save As...* buttons instead.

Close

The *Close* button closes the *HTMLDOC* window.

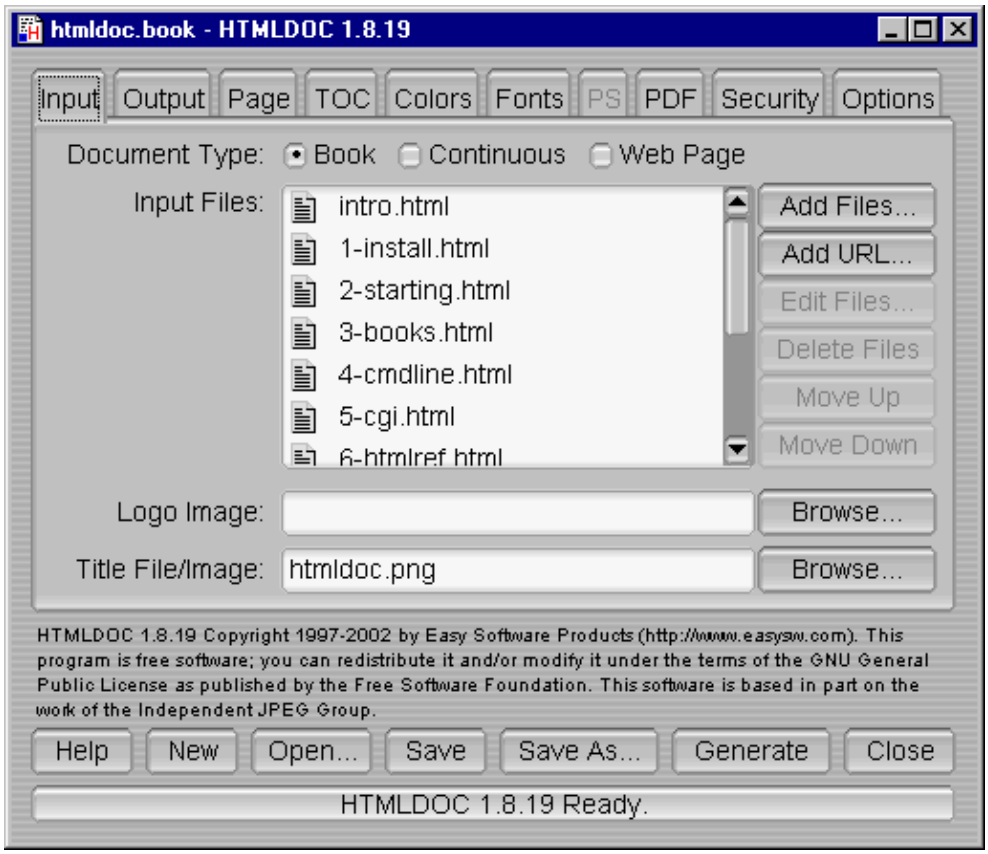

Figure 7−1 − The Input Tab

The Input Tab

The input tab (Figure 7−1) lists all of the HTML source files that are used to generate the document. You also specify the type of document (book or web page) and the title and logo images in this tab.

Document Type

The *Book* radio button specifies that the input files are structured with headings. The *Continuous* radio button specifies unstructured files without page breaks between each file. The *Web Page* radio button specifies unstructured files with page breaks between each file.

Input Files

The *Input Files* list shows all of the HTML input files that will be used to produce the document. Double−click on files to edit them.

Add Files...

The *Add Files...* button displays the file chooser dialog, allowing you to select one or more HTML files to include in the document.

Edit Files...

The *Edit Files...* button starts the specified editor program to edit the files selected in the *Input Files* list. Select one or more files in the *Input Files* list to enable the *Edit Files...* button.

Delete Files

The *Delete Files* button removes the selected files from the *Input Files* list. Select one or more files in the *Input Files* list to enable the *Delete Files* button.

The *Delete Files* button only removes the files from the *Input Files* list. The files are ***not*** removed from disk.

Move Up

The *Move Up* button moves the selected files in the *Input Files* list up one line in the list. To enable the *Move Up* button select one or more files in the *Input Files* list.

Move Down

The *Move Down* button moves the selected files in the *Input Files* list down one line in the list. To enable the *Move Down* button select one or more files in the *Input Files* list.

Logo Image

The *Logo Image* field contains the filename for an image to be shown in the header or footer of pages, and in the navigation bar of HTML files.

Click on the *Browse...* button to select a logo image file using the file chooser dialog.

Title File/Image

The *Title File/Image* field contains the filename for an image to be shown on the title page, or for a HTML file to be used for the title page(s).

Click on the *Browse...* button to select a title file using the file chooser dialog.

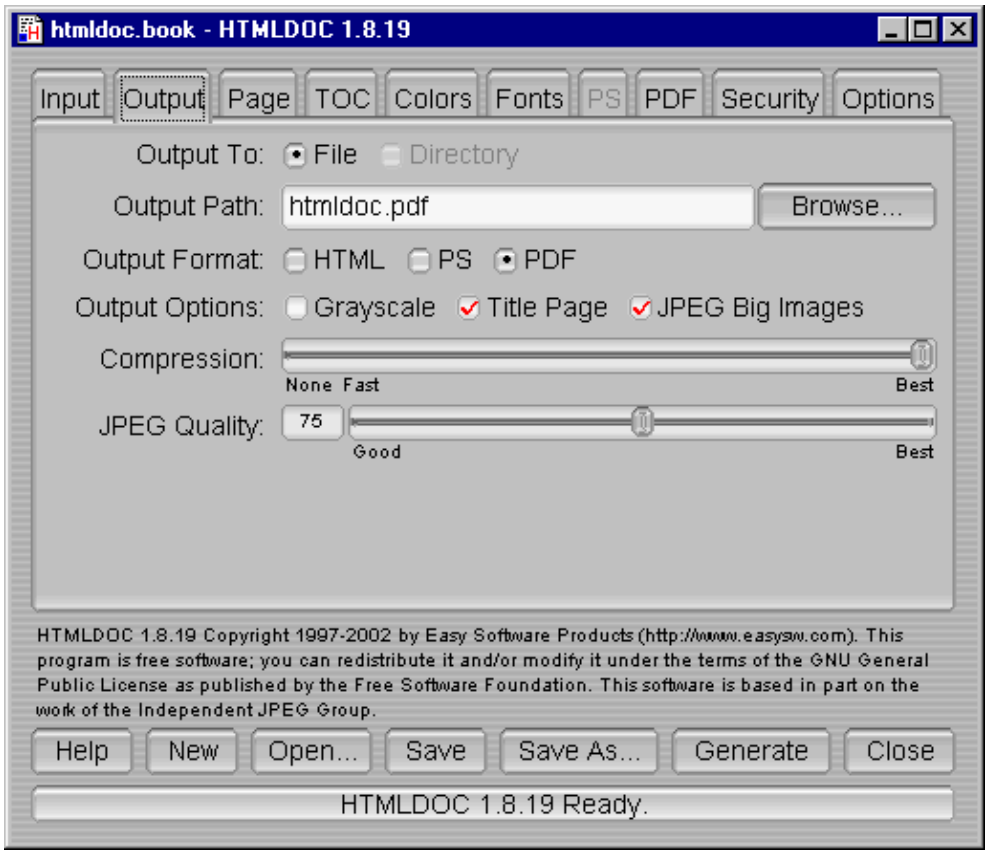

Figure 7−2 − The Output Tab

The Output Tab

The output tab (Figure 7−2) specifies where your document will be generated, the output format, and some of the generic output options.

Output To

The *File* radio button selects output to a single file. The *Directory* radio button selects output to multiple files in the named directory.

Directory output is not available when generating PDF files.

Output Path

The *Output Path* field contains the output directory or filename. Click on the *Browse...* button to choose an output file using the file chooser dialog.

Output Format

The *HTML* radio button selects HTML output, the *PS* radio button selects PostScript output, and the *PDF* radio button selects PDF output.

Output Options

The *Grayscale* check box selects grayscale output for PostScript and PDF files. The *Title Page* check box specifies that a title page should be generated for the document. The *JPEG Big Images* check box specifies that JPEG compression should be applied to continuous−tone images.

Compression

The *Compression* slider controls the amount of compression that is used when writing PDF or Level 3 PostScript output.

Note: *HTMLDOC* uses Flate compression, which is not encumbered by patents and is also used by the popular PKZIP and gzip programs. Flate is a lossless compression algorithm (that is, you get back exactly what you put in) that performs very well on indexed images and text.

JPEG Quality

The *JPEG Quality* slider controls the quality level used when writing continuous−tone images with JPEG compression.

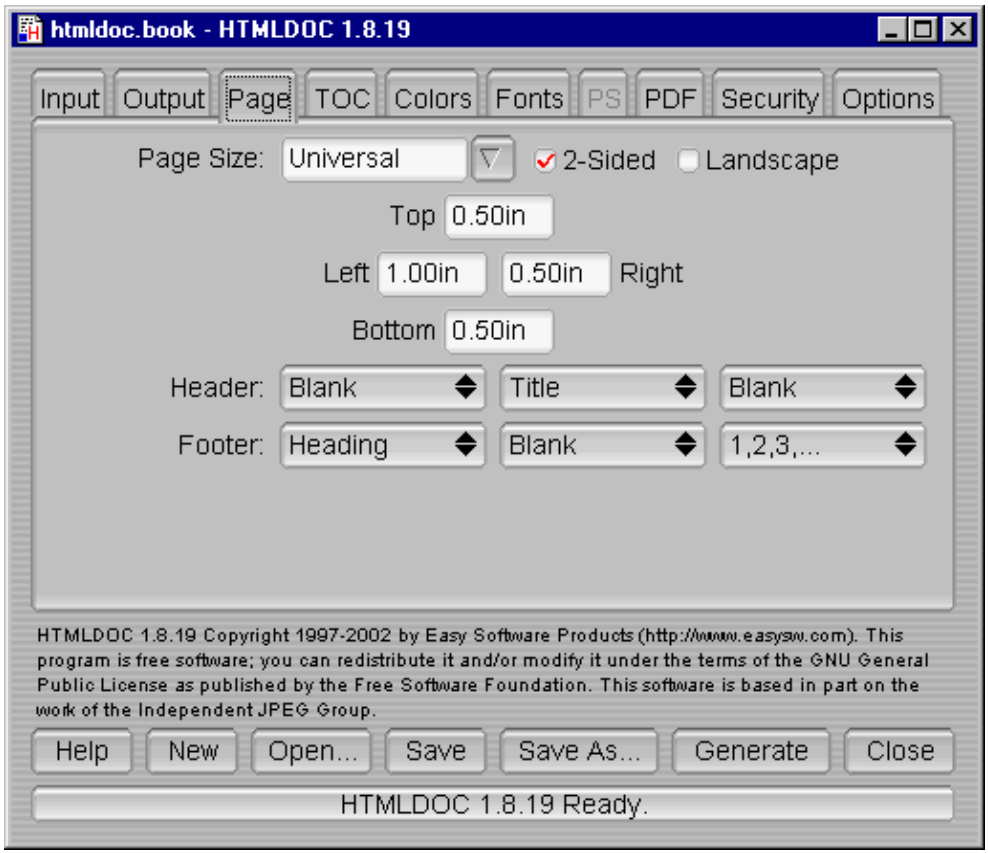

Figure 7−3 − The Page Tab

The Page Tab

The page tab (Figure 7−3) defines the page header, footer, size, and margins for PostScript and PDF output.

Page Size

The *Page Size* field contains the current page size. Click on the arrow button to choose a standard page size.

HTMLDOC supports the following standard page size names:

- Letter − 8.5x11in (216x279mm)
- A4 − 8.27x11.69in (210x297mm)
- Universal − 8.27x11in (210x279mm)

Click in the *Page Size* field and enter the page width and length separated by the letter "x" to select a custom page size. Append the letters "in" for inches, "mm" for millimeters, or "cm" for centimeters.

2−Sided

Click in the *2−Sided* check box to select 2−sided (duplexed) output.

Landscape

Click in the *Landscape* check box to select landscape output.

Top, Left, Right, and Bottom

Click in the *Top*, *Left*, *Right*, and *Bottom* fields and enter the new margin values to change them. Append the letters "in" for inches, "mm" for millimeters, or "cm" for centimeters.

Header and Footer

Select the desired text in each of the option buttons to customize the header and footer for the document/body pages. The left−most option buttons set the text that is left−justified, while the middle buttons set the text that is centered and the right buttons set the text that is right−justified. Each choice corresponds to the following text:

Choice	Description
Blank	The field should be blank.
Title	The field should contain the document title.
Chapter Title	The field should contain the current chapter title.
Heading	The field should contain the current heading.
Logo	The field should contain the logo image.
1,2,3,...	The field should contain the current page number in decimal format (1, 2, 3, ...)
i,ii,iii,...	The field should contain the current page number in lowercase roman numerals (i, ii, iii, ...)
I,II,III,...	The field should contain the current page number in uppercase roman numerals (I, II, III, ...)
a,b,c,...	The field should contain the current page number using lowercase letters.
A,B,C,...	The field should contain the current page number using UPPERCASE letters.
Chapter Page	The field should contain the current chapter page number.
1/N,2/N,...	The field should contain the current and total number of pages (n/N).
1/C,2/C,...	The field should contain the current and total number of pages in the chapter (n/N).
Date	The field should contain the current date (formatted for the current locale).
Time	The field should contain the current time (formatted for the current locale).
Date + Time	The field should contain the current date and time (formatted for the current locale).

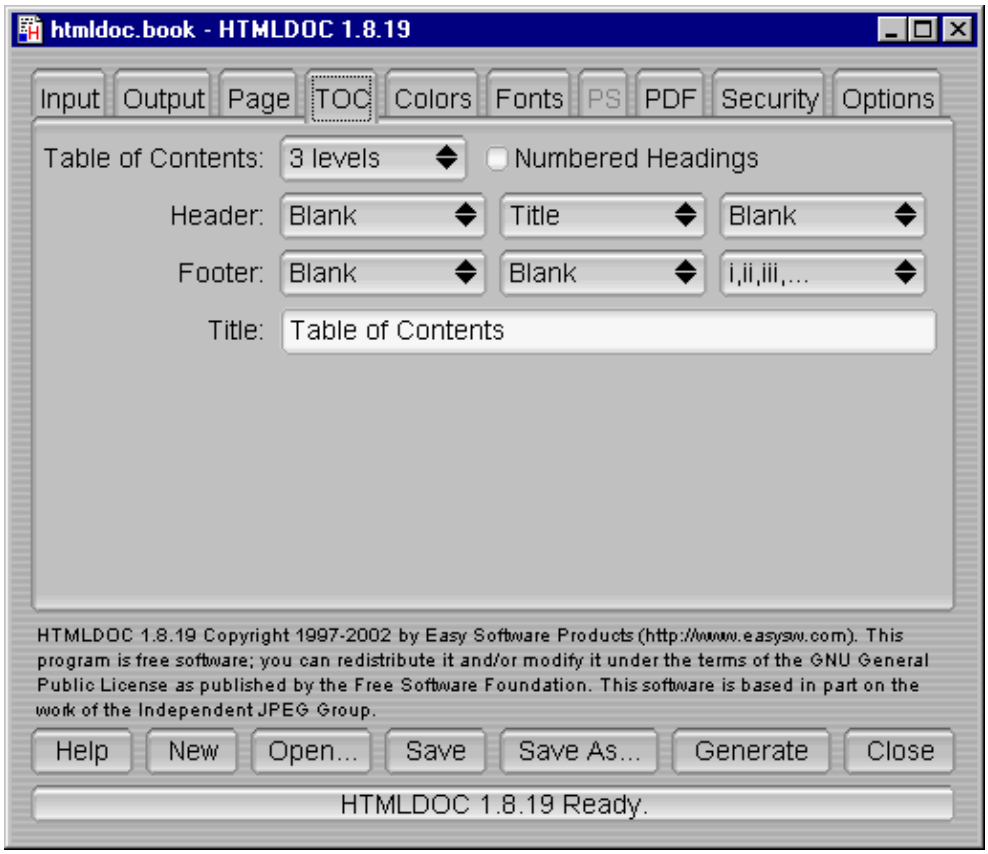

Figure 7−4 − The TOC Tab

The TOC Tab

The TOC tab (Figure 7−4) defines the table−of−contents options.

Table of Contents

Select the desired number of levels from the *Table of Contents* option button.

Numbered Headings

Click in the *Numbered Headings* check box to automatically number the headings in the document.

Header and Footer

Select the desired text in each of the option buttons to customize the header and footer for the tables−of−contents pages. The left−most option buttons set the text that is left−justified, while the middle buttons set the text that is centered and the right buttons set the text that is right−justified.

Title

Enter the desired title for the table−of−contents in the *Title* field.

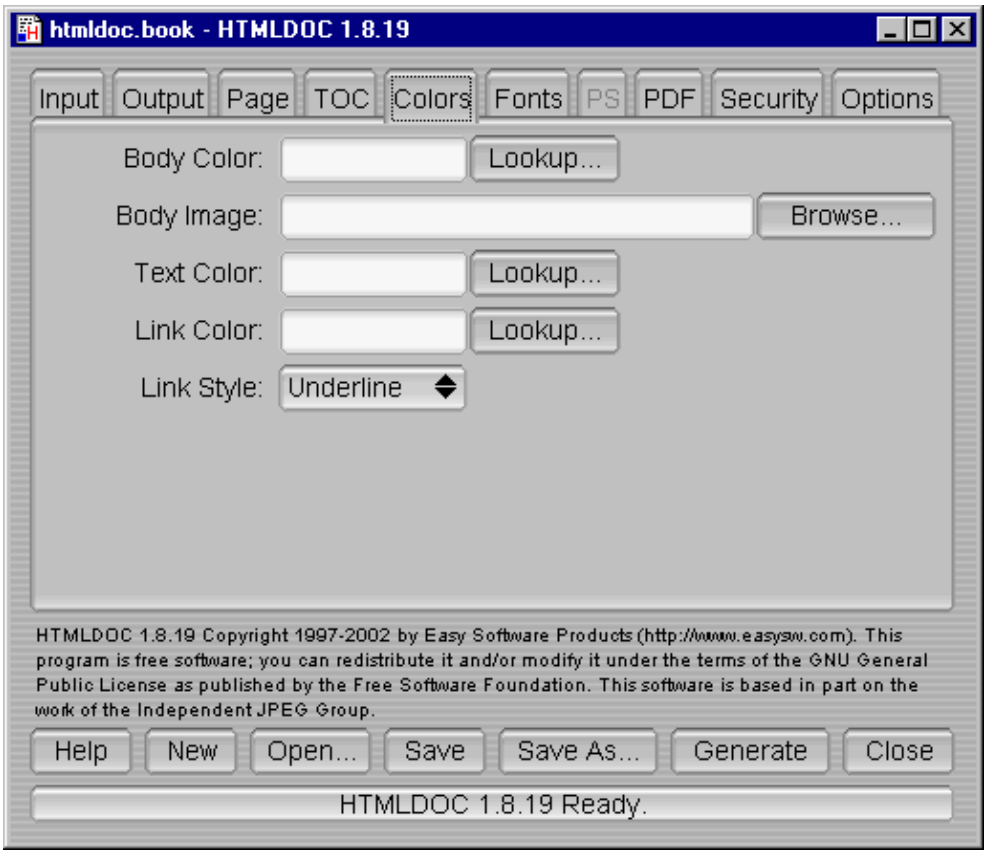

Figure 7−5 − The Colors Tab

The Colors Tab

The colors tab (Figure 7−5) defines the color and image information that is used for the entire document.

Body Color

The *Body Color* field specifies the default background color. It can be a standard HTML color name or a hexadecimal RGB color of the form #RRGGBB. Click on the *Lookup...* button to pick the color graphically.

Body Image

The *Body Image* field specifies the default background image. Click on the *Browse...* button to pick the background image using the file chooser.

Text Color

The *Text Color* field specifies the default text color. It can be a standard HTML color name or a hexadecimal RGB color of the form #RRGGBB. Click on the *Lookup...* button to pick the color graphically.

Link Color

The *Link Color* field specifies the default link color. It can be a standard HTML color name or a hexadecimal RGB color of the form #RRGGBB. Click on the *Lookup...* button to pick the color graphically.

Link Style

The *Link Style* chooser specifies the default link decoration.

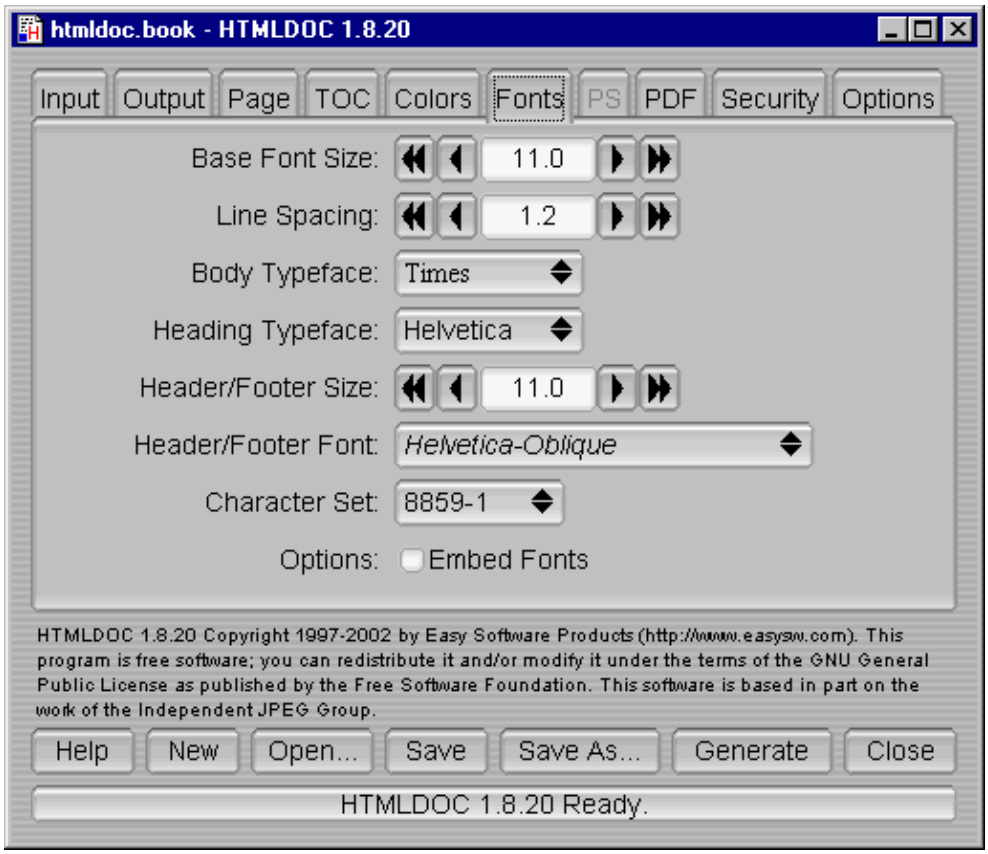

Figure 7−6 − The Fonts Tab

The Fonts Tab

The fonts tab (Figure 7−6) defines the fonts and character set used by the document.

Base Font Size

The *Base Font Size* field specifies the size of normal text in the document in points (1 point = 1/72nd inch). Click on the single arrow buttons to decrease or increase the size by 1/10th point or on the double arrow buttons to decrease or increase the size by whole points.

Line Spacing

The *Line Spacing* field specifies the spacing between lines as a multiple of the base font size. Click on the single arrow buttons to decrease or increase the size by 10ths or on the double arrow buttons to decrease or increase the size by whole numbers.

Body Typeface

The *Body Typeface* option button specifies the typeface to use for normal text. Click on the option button to

select a typeface.

Heading Typeface

The *Heading Typeface* option button specifies the typeface to use for headings. Click on the option button to select a typeface.

Header/Footer Size

The *Header/Footer Size* field specifies the size of header and footer text in the document in points (1 point = 1/72nd inch). Click on the single arrow buttons to decrease or increase the size by 1/10th point or on the double arrow buttons to decrease or increase the size by whole points.

Header/Footer Font

The *Header/Footer Font* option button specifies the typeface and style to use for header and footer text. Click on the option button to select a typeface and style.

Character Set

The *Character Set* option button specifies the encoding of characters in the document. Click on the option button to select a character set.

Options

The *Embed Fonts* check box controls whether or not fonts are embedded in PostScript and PDF output.

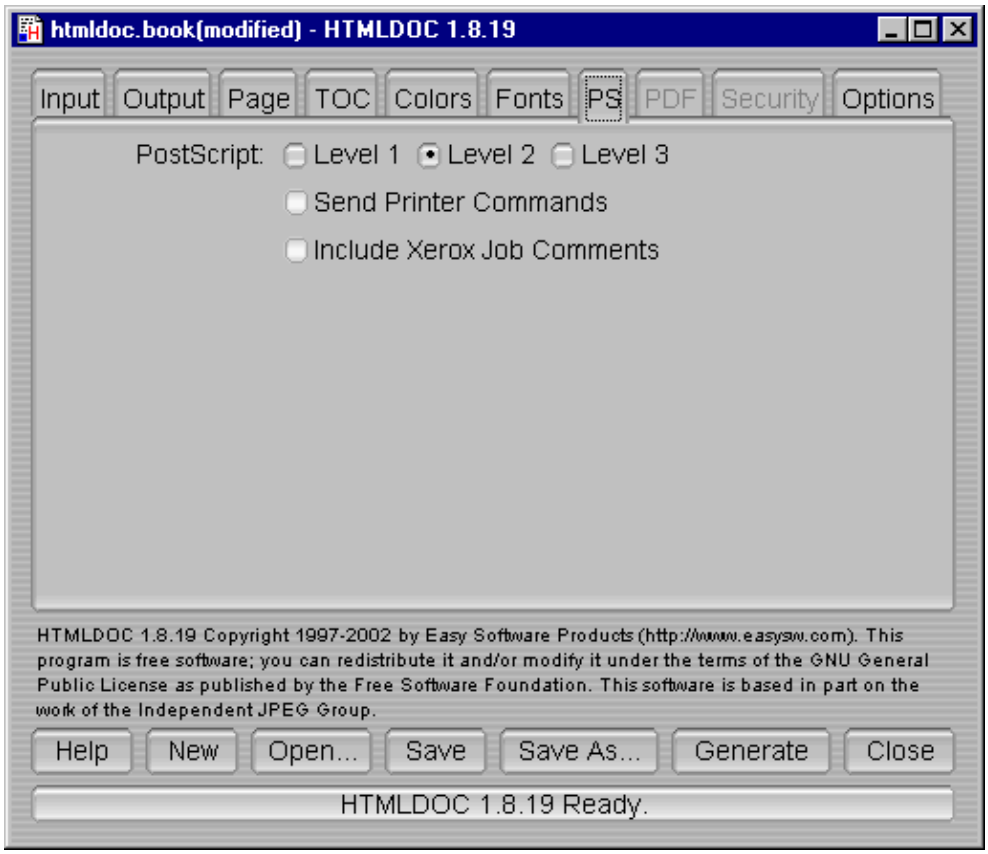

Figure 7−7 − The PS Tab

The PS Tab

The PS tab (Figure 7−7) contains options specific to PostScript output.

PostScript Level

Click on one of the *Level* radio buttons to select the language level to generate. PostScript Level 1 is compatible with all PostScript printers and will produce the largest output files.

PostScript Level 2 is compatible with most PostScript printers and supports printer commands and JPEG image compression.

PostScript Level 3 is compatible with only the newest PostScript printers and supports Flate image compression in addition to the Level 2 features.

Send Printer Commands

The *Send Printer Commands* check box controls whether or not the output files contain PostScript `setpagedevice` commands for the page size and duplex settings. Click in the check box to enable or disable printer commands.

Printer commands are only available with Level 2 and 3 output and may not work with some printers.

Include Xerox Job Comments

The *Include Xerox Job Comments* check box controls whether or not the output files contain Xerox job comments. Click in the check box to enable or disable the job comments.

Job comments are available with all levels of PostScript output.

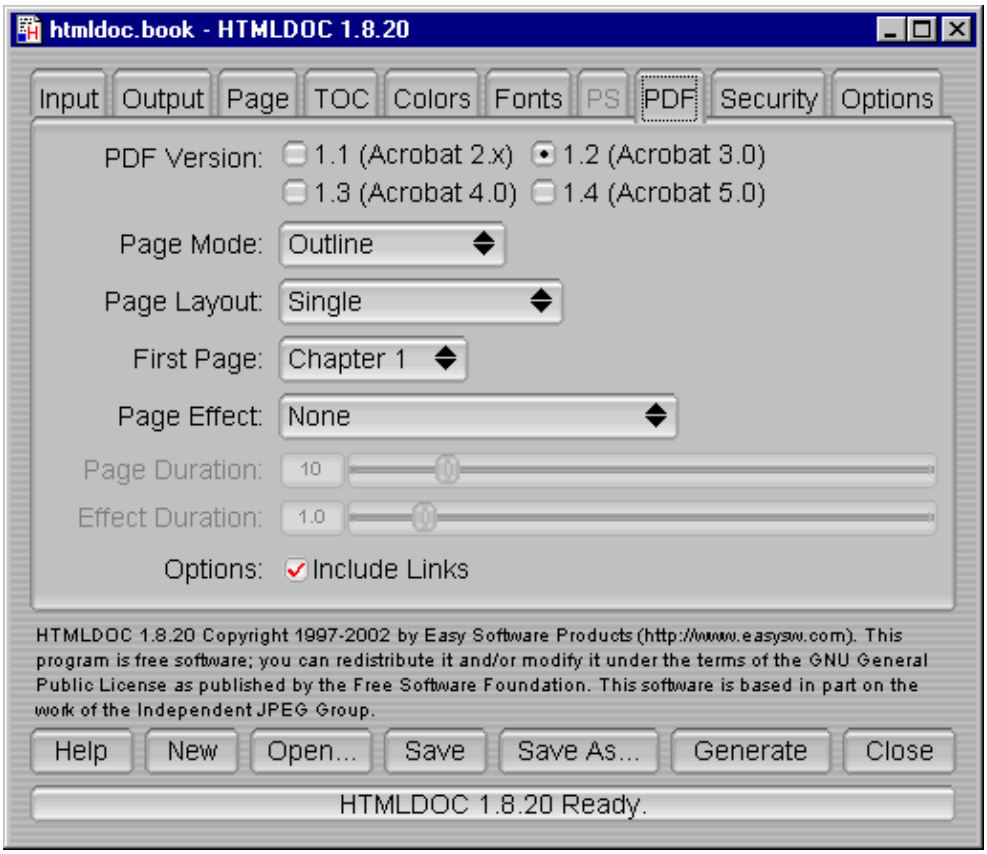

Figure 7−8 − The PDF Tab

The PDF Tab

The PDF tab (Figure 7−8) contains settings specific to PDF output.

PDF Version

The *PDF Version* radio buttons control what version of PDF is generated. PDF 1.3 is the most commonly supported version. Click on the corresponding radio button to set the version.

Page Mode

The *Page Mode* option button controls the initial viewing mode for the document. Click on the option button to set the page mode.

The *Document* page mode displays only the document pages. The *Outline* page mode displays the table−of−contents outline as well as the document pages. The *Full−Screen* page mode displays the document pages on the whole screen; this mode is used primarily for presentations.

Page Layout

The *Page Layout* option button controls the initial layout of document pages on the screen. Click on the option button to set the page layout.

The *Single* page layout displays a single page at a time. The *One Column* page layout displays a single column of pages at a time. The *Two Column Left* and *Two Column Right* page layouts display two columns of pages at a time; the first page is displayed in the left or right column as selected.

First Page

The *First Page* option button controls the initial page that is displayed. Click on the option button to choose the first page.

Page Effect

The *Page Effect* option button controls the page effect that is displayed in *Full−Screen* mode. Click on the option button to select a page effect.

Page Duration

The *Page Duration* slider controls the number of seconds that each page will be visible in *Full−Screen* mode. Drag the slider to adjust the number of seconds.

Effect Duration

The *Effect Duration* slider controls the number of seconds that the page effect will last when changing pages. Drag the slider to adjust the number of seconds.

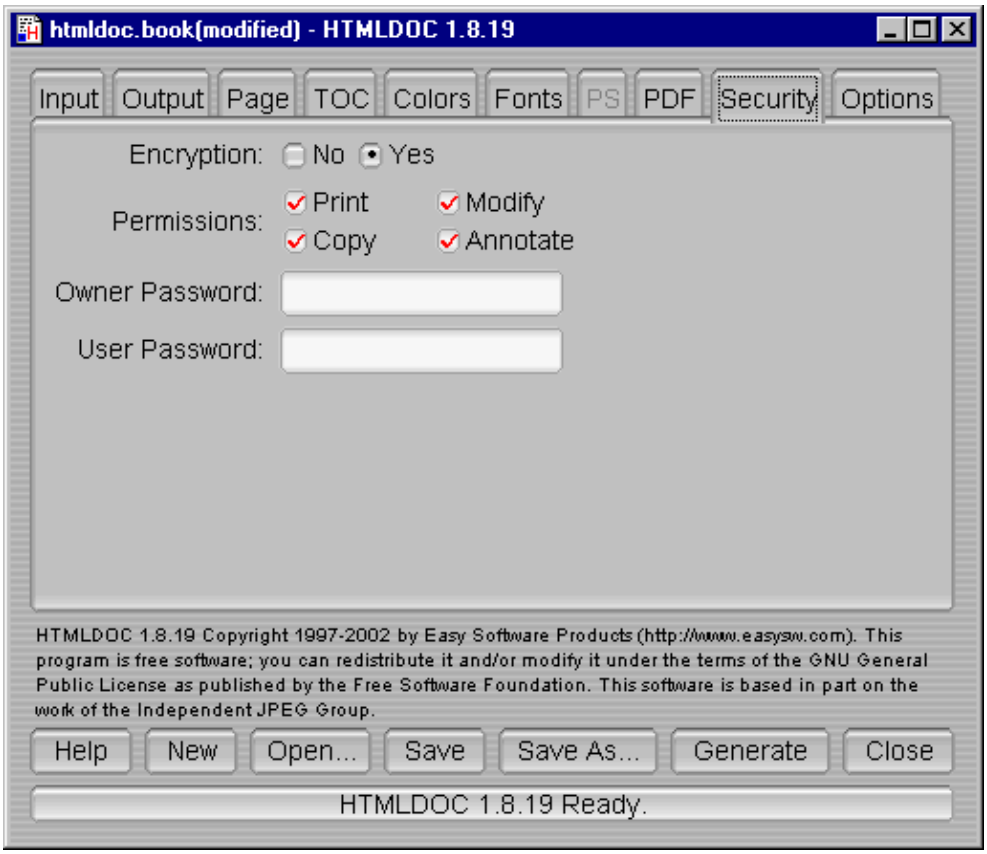

Figure 7−9 − The Security Tab

The Security Tab

The security tab (Figure 7−9) allows you to enable PDF document encryption and security features.

Encryption

The *Encryption* buttons control whether or not encryption is performed on the PDF file. Encrypted documents can be password protected and also provide user permissions.

Permissions

The *Permissions* buttons control what operations are allowed by the PDF viewer.

Owner Password

The *Owner Password* field contains the document owner password, a string that is used by Adobe Acrobat to control who can change document permissions, etc.

If this field is left blank, a random 32−character password is generated so that no one can change the document using the Adobe tools.

User Password

The *User Password* field contains the document user password, a string that is used by Adobe Acrobat to restrict viewing permissions on the file.

If this field is left blank, any user may view the document without entering a password.

Options

The *Include Links* option controls whether or not the internal links in a document are included in the PDF output. The document outline (shown to the left of the document in Acrobat Reader) is unaffected by this setting.

User Password

The *User Password* field contains the document user password, a string that is used by Adobe Acrobat to restrict viewing permissions on the file.

If this field is left blank, any user may view the document without entering a password.

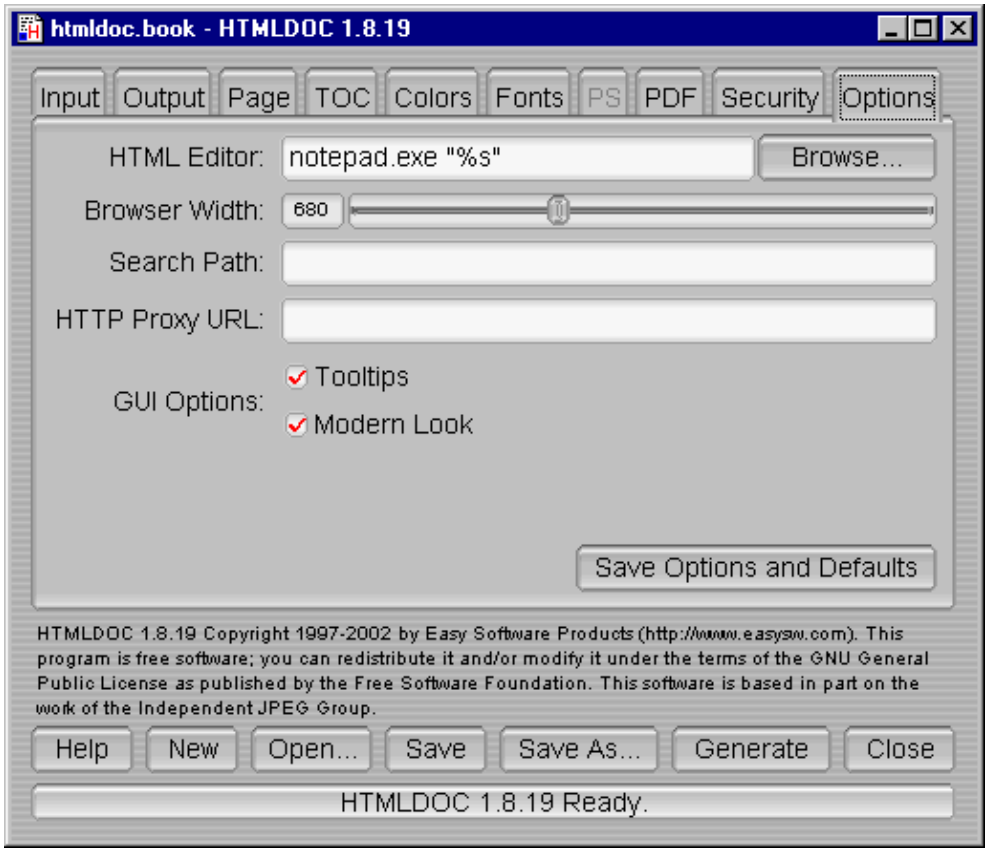

Figure 7−10 − The Options Tab

The Options Tab

The options tab (Figure 7−10) contains the HTML file editor of your choice and allows you to save the settings and options that will be used in new documents.

HTML Editor

The *HTML Editor* field contains the name of the HTML editor to run when you double−click on an input file or click on the *Edit Files...* button. Enter the program name in the field or click on the *Browse...* button to select the editor using the file chooser.

The `%s` is added automatically to the end of the command name to insert the name of the file to be edited. If you are using Netscape Composer to edit your HTML files you should put "−edit" before the `%s` to tell Netscape to edit the file and not display it.

Browser Width

The *Browser Width* slider specifies the width of the browser in pixels that is used to scale images and other pixel measurements to the printable page width. You can adjust this value to more closely match the formatting on the screen.

The default browser width is 680 pixels which corresponds roughly to a 96 DPI display. The browser width is only used when generating PostScript or PDF files.

Search Path

The *Search Path* field specifies a search path for files that are loaded by HTMLDOC. It is usually used to get images that use absolute server paths to load.

Directories are separated by the semicolon (;) so that drive letters (and eventually URLs) can be specified.

Proxy URL

The *Proxy URL* field specifies a URL for a HTTP proxy server.

Tooltips

The *Tooltips* check button controls the appearance of tooltip windows over GUI controls.

Modern Look

The *Modern Look* check button controls the appearance of the GUI controls.

Strict HTML

The *Strict HTML* check button controls strict HTML conformance checking. When checked, HTML elements that are improperly nested and dangling close elements will produce error messages.

Save Options and Defaults

The *Save Options and Defaults* button saves the HTML editor and all of the document settings on the other tabs for use in new documents. These settings are also used by the command−line version of *HTMLDOC*.

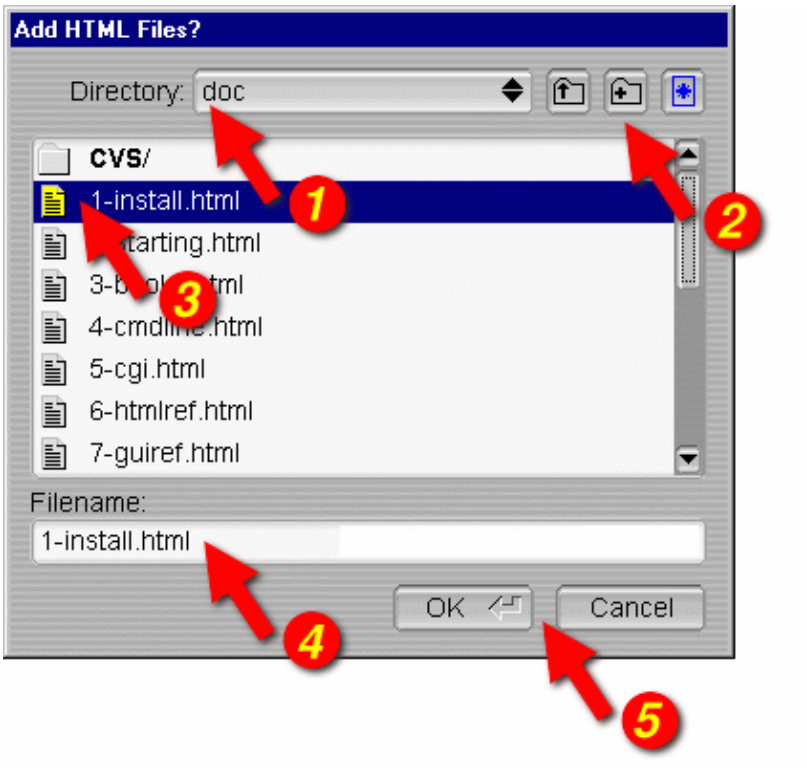

Figure 7−11 − The File Chooser

The File Chooser

The file chooser (Figure 7−11) allows you to select one or more files and create files and directories.

Directory

The *Directory* option button (1) shows the current directory or folder that is displayed in the file list (3). Click on the option button to navigate to other directories or folders.

Directory Buttons

The directory buttons (2) allow you to go up one level in the directory hierarchy, create a new directory, and change the filename filter settings, respectively.

File List

The file list (3) lists the files and directories in the current directory or folder. Double−click on a file or directory to select that file or directory. Drag the mouse or hold the **CTRL** key down while clicking to select multiple files.

Filename

The *Filename* field contains the currently selected filename. Type a name in the field to select a file or directory. As you type, any matching filenames will be highlighted; press the **TAB** key to accept the matches.

Dialog Buttons

The dialog buttons (5) close the file chooser dialog window. Click on the *OK* button to accept your selections or the *Cancel* button to reject your selections and cancel the file operation.

Chapter 8 − Command−Line Reference

This chapter describes all of the command−line options supported by *HTMLDOC*.

> **Note:**
>
> The free version of *HTMLDOC* for Windows does not include the command−line program.

Basic Usage

The basic command−line usage for *HTMLDOC* is:

```
% htmldoc options filename1.html ... filenameN.html ENTER
% htmldoc options filename.book ENTER
```

The first form converts the named HTML files to the specified output format immediately. The second form loads the specified `.book` file and displays the *HTMLDOC* window, allowing a user to make changes and/or generate the document interactively.

If no output file or directory is specified, then all output is sent to the standard output file.

Options

The following command−line options are recognized by *HTMLDOC*.

−d directory

The `−d` option specifies an output directory for the document files.

This option is not compatible with the PDF output format.

−f filename

The `−f` option specifies an output file for the document.

−t format

The `−t` option specifies the output format for the document and can be one of the following:

Format	Description
html	Generate one or more indexed HTML files.
pdf	Generate a PDF file (default version − 1.3).
pdf11	Generate a PDF 1.1 file for Acrobat Reader 2.0.
pdf12	Generate a PDF 1.2 file for Acrobat Reader 3.0.
pdf13	Generate a PDF 1.3 file for Acrobat Reader 4.0.
pdf14	Generate a PDF 1.4 file for Acrobat Reader 5.0.
ps	Generate one or more PostScript files (default level).
ps1	Generate one or more Level 1 PostScript files.
ps2	Generate one or more Level 2 PostScript files.
ps3	Generate one or more Level 3 PostScript files.

−v

The `−v` option specifies that progress information should be sent/displayed to the standard error file.

−−batch filename.book

The `−−batch` option specifies a book file that you would like to generate without the GUI popping up. This option can be combined with other options to generate the same book in different formats and sizes:

```
% htmldoc −−batch filename.book −f filename.ps ENTER
% htmldoc −−batch filename.book −f filename.pdf ENTER
```

--bodycolor color

The `--bodycolor` option specifies the background color for all pages in the document. The color can be specified by a standard HTML color name or as a 6-digit hexadecimal number of the form `#RRGGBB`.

--bodyfont typeface

The `--bodyfont` option specifies the default text font used for text in the document body. The `typeface` parameter can be one of the following:

typeface	Actual Font
Arial	Helvetica
Courier	Courier
Helvetica	Helvetica
Monospace	Courier
Sans-Serif	Helvetica
Serif	Times
Symbol	Symbol
Times	Times

--bodyimage filename

The `--bodyimage` option specifies the background image for all pages in the document. The supported formats are BMP, GIF, JPEG, and PNG.

--book

The `--book` option specifies that the input files comprise a book with chapters and headings.

--bottom margin

The `--bottom` option specifies the bottom margin. The default units are points (1 point = 1/72nd inch); the suffixes "in", "cm", and "mm" specify inches, centimeters, and millimeters, respectively.

This option is only available when generating PostScript or PDF files.

--browserwidth pixels

The `--browserwidth` option specifies the browser width in pixels. The browser width is used to scale images and pixel measurements when generating PostScript and PDF files. It does not affect the font size of text.

The default browser width is 680 pixels which corresponds roughly to a 96 DPI display. Please note that your images and table sizes are equal to or smaller than the browser width, or your output will overlap or truncate

in places.

--charset charset

The `--charset` option specifies the 8-bit character set encoding to use for the entire document. *HTMLDOC* comes with the following character set files:

charset	Character Set
iso-8859-1	ISO-8859-1
iso-8859-2	ISO-8859-2
iso-8859-3	ISO-8859-3
iso-8859-4	ISO-8859-4
iso-8859-5	ISO-8859-5
iso-8859-6	ISO-8859-6
iso-8859-7	ISO-8859-7
iso-8859-8	ISO-8859-8
iso-8859-9	ISO-8859-9
iso-8859-14	ISO-8859-14
iso-8859-15	ISO-8859-15
koi8-r	KOI8-R

--color

The `--color` option specifies that color output is desired.

This option is only available when generating PostScript or PDF files.

--compression[=level]

The `--compression` option specifies that Flate compression should be performed on the output file(s). The optional `level` parameter is a number from 1 (fastest and least amount of compression) to 9 (slowest and most amount of compression).

This option is only available when generating Level 3 PostScript or PDF files.

--continuous

The `--continuous` option specifies that the input files comprise a web page (or site) and that no title page or table-of-contents should be generated. Unlike the `--webpage` option described later in this chapter, page breaks are not inserted between each input file.

This option is only available when generating PostScript or PDF files.

--datadir directory

The `--datadir` option specifies the location of data files used by *HTMLDOC*.

--duplex

The `--duplex` option specifies that the output should be formatted for two sided printing.

This option is only available when generating PostScript or PDF files. Use the `--pscommands` option to generate PostScript duplex mode commands.

--effectduration seconds

The `--effectduration` option specifies the duration of a page transition effect in seconds.

This option is only available when generating PDF files.

--embedfonts

The `--embedfonts` option specifies that fonts should be embedded in PostScript and PDF output. This is especially useful when generating documents in character sets other than ISO−8859−1.

--encryption

The `--encryption` option enables encryption and security features for PDF output.

This option is only available when generating PDF files.

--firstpage page

The `--firstpage` option specifies the first page that will be displayed in a PDF file. The `page` parameter can be one of the following:

page	Description
p1	The first page of the document.
toc	The first page of the table-of-contents.
c1	The first page of chapter 1.

This option is only available when generating PDF files.

--fontsize size

The `--fontsize` option specifies the base font size for the entire document in points (1 point = 1/72nd inch).

--fontspacing spacing

The `--fontspacing` option specifies the line spacing for the entire document as a multiplier of the base font size. A `spacing` value of 1 makes each line of text the same height as the font.

--footer lcr

The `--footer` option specifies the contents of the page footer. The `lcr` parameter is a three-character string representing the left, center, and right footer fields. Each character can be one of the following:

lcr	Description
.	A period indicates that the field should be blank.
:	A colon indicates that the field should contain the current and total number of pages in the chapter (n/N).
/	A slash indicates that the field should contain the current and total number of pages (n/N).
1	The number 1 indicates that the field should contain the current page number in decimal format (1, 2, 3, ...)
a	A lowercase "a" indicates that the field should contain the current page number using lowercase letters.
A	An uppercase "A" indicates that the field should contain the current page number using UPPERCASE letters.
c	A lowercase "c" indicates that the field should contain the current chapter title.
C	An uppercase "C" indicates that the field should contain the current chapter page number.
d	A lowercase "d" indicates that the field should contain the current date.
D	An uppercase "D" indicates that the field should contain the current date and time.
h	An "h" indicates that the field should contain the current heading.
i	A lowercase "i" indicates that the field should contain the current page number in lowercase roman numerals (i, ii, iii, ...)
I	An uppercase "I" indicates that the field should contain the current page number in uppercase roman numerals (I, II, III, ...)
l	A lowercase "l" indicates that the field should contain the logo image.
t	A lowercase "t" indicates that the field should contain the document title.
T	An uppercase "T" indicates that the field should contain the current time.

Setting the footer to "`...`" disables the footer entirely.

--format format

The `--format` option specifies the output format for the document and can be one of the following:

Format	Description
html	Generate one or more indexed HTML files.
pdf	Generate a PDF file (default version − 1.3).
pdf11	Generate a PDF 1.1 file for Acrobat Reader 2.0.
pdf12	Generate a PDF 1.2 file for Acrobat Reader 3.0.
pdf13	Generate a PDF 1.3 file for Acrobat Reader 4.0.
pdf14	Generate a PDF 1.4 file for Acrobat Reader 5.0.
ps	Generate one or more PostScript files (default level).
ps1	Generate one or more Level 1 PostScript files.
ps2	Generate one or more Level 2 PostScript files.
ps3	Generate one or more Level 3 PostScript files.

--gray

The `--gray` option specifies that grayscale output is desired.

This option is only available when generating PostScript or PDF files.

--header lcr

The `--header` option specifies the contents of the page header. The `lcr` parameter is a three−character string representing the left, center, and right header fields. See the `--footer` option for the list of formatting characters.

Setting the header to "`...`" disables the header entirely.

--headfootfont font

The `--headfootfont` option specifies the font that is used for the header and footer text. The `font` parameter can be one of the following:

- Courier
- Courier-Bold
- Courier-Oblique
- Courier-BoldOblique
- Times
- Times-Roman
- Times-Bold
- Times-Italic
- Times-BoldItalic
- Helvetica
- Helvetica-Bold
- Helvetica-Oblique
- Helvetica-BoldOblique

This option is only available when generating PostScript or PDF files.

--headfootsize size

The `--headfootsize` option sets the size of the header and footer text in points (1 point = 1/72nd inch).

This option is only available when generating PostScript or PDF files.

--headingfont typeface

The `--headingfont` options sets the typeface that is used for headings in the document. The `typeface` parameter can be one of the following:

typeface	Actual Font
Arial	Helvetica
Courier	Courier
Helvetica	Helvetica
Monospace	Courier
Sans-Serif	Helvetica
Serif	Times
Symbol	Symbol
Times	Times

--help

The `--help` option displays all of the available options to the standard output file.

--helpdir directory

The `--helpdir` option specifies the location of the on-line help files.

--jpeg[=quality]

The `--jpeg` option enables JPEG compression of continuous-tone images. The optional `quality` parameter specifies the output quality from 0 (worst) to 100 (best).

This option is only available when generating Level 2 and Level 3 PostScript or PDF files.

--landscape

The `--landscape` option specifies that the output should be in landscape orientation (long edge on top).

This option is only available when generating PostScript or PDF files.

--left margin

The `--left` option specifies the left margin. The default units are points (1 point = 1/72nd inch); the suffixes "in", "cm", and "mm" specify inches, centimeters, and millimeters, respectively.

This option is only available when generating PostScript or PDF files.

--linkcolor color

The `--linkcolor` option specifies the color of links in HTML and PDF output. The color can be specified by name or as a 6-digit hexadecimal number of the form `#RRGGBB`.

--links

The `--links` option specifies that PDF output should contain hyperlinks.

--linkstyle style

The `--linkstyle` option specifies the style of links in HTML and PDF output. The style can be "plain" for no decoration or "underline" to underline links.

--logoimage filename

The `--logoimage` option specifies the logo image for the HTML navigation bar and page headers and footers for PostScript and PDF files. The supported formats are BMP, GIF, JPEG, and PNG.

--no-compression

The `--no-compression` option specifies that Flate compression should not be performed on the output files.

--no-duplex

The `--no-duplex` option specifies that the output should be formatted for one sided printing.

This option is only available when generating PostScript or PDF files. Use the `--pscommands` option to generate PostScript duplex mode commands.

--no-embedfonts

The `--no-embedfonts` option specifies that fonts should not be embedded in PostScript and PDF output.

--no-encryption

The `--no-encryption` option specifies that no encryption/security features should be enabled in PDF output.

This option is only available when generating PDF files.

--no-jpeg

The `--no-jpeg` option specifies that JPEG compression should not be performed on large images.

--no-links

The `--no-links` option specifies that PDF output should not contain hyperlinks.

--no-localfiles

The `--no-localfiles` option disables access to local files on the system. This option should be used when providing remote document conversion services.

--no-numbered

The `--no-numbered` option specifies that headings should not be numbered.

--no-pscommands

The `--no-pscommands` option specifies that PostScript device commands should not be written to the output files.

--no-strict

The `--no-strict` option turns off strict HTML conformance checking.

--no-title

The `--no-title` option specifies that the title page should not be generated.

--no-toc

The `--no-toc` option specifies that the table-of-contents pages should not be generated.

--no-xrxcomments

The `--no-xrxcomments` option specifies that Xerox PostScript job comments should not be written to the output files.

This option is only available when generating PostScript files.

--numbered

The `--numbered` option specifies that headings should be numbered.

--outdir directory

The `--outdir` option specifies an output directory for the document files.

This option is not compatible with the PDF output format.

--outfile filename

The `--outfile` option specifies an output file for the document.

--owner-password password

The `--owner-password` option specifies the owner password for a PDF file. If not specified or the empty string (""), a random password is generated.

This option is only available when generating PDF files.

--pageduration seconds

The `--pageduration` option specifies the number of seconds that each page will be displayed in the document.

This option is only available when generating PDF files.

--pageeffect effect

The `--pageeffect` option specifies the page effect to use in PDF files. The `effect` parameter can be one of the following:

effect	Description
none	No effect is generated.
bi	Box Inward
bo	Box Outward
d	Dissolve
gd	Glitter Down
gdr	Glitter Down and Right
gr	Glitter Right
hb	Horizontal Blinds
hsi	Horizontal Sweet Inward
hso	Horizontal Sweep Outward
vb	Vertical Blinds
vsi	Vertical Sweep Inward
vso	Vertical Sweep Outward
wd	Wipe Down
wl	Wipe Left
wr	Wipe Right
wu	Wipe Up

This option is only available when generating PDF files.

--pagelayout layout

The `--pagelayout` option specifies the initial page layout in the PDF viewer. The `layout` parameter can be one of the following:

layout	Description
single	A single page is displayed.
one	A single column is displayed.
twoleft	Two columns are displayed with the first page on the left.
tworight	Two columns are displayed with the first page on the right.

This option is only available when generating PDF files.

--pagemode mode

The `--pagemode` option specifies the initial viewing mode in the PDF viewer. The `mode` parameter can be one of the following:

mode	Description
document	The document pages are displayed in a normal window.
outline	The document outline and pages are displayed.
fullscreen	The document pages are displayed on the entire screen in "slideshow" mode.

This option is only available when generating PDF files.

--path dir1;dir2;dir3;...;dirN

The `--path` option specifies a search path for files that are loaded by HTMLDOC. It is usually used to get images that use absolute server paths to load.

Directories are separated by the semicolon (;) so that drive letters and URLs can be specified. Quotes around the directory parameter are optional. They are usually used when the directory string contains spaces.

```
--path "dir1;dir2;dir3;...;dirN"
```

--permissions permission

The `--permissions` option specifies the document permissions. Multiple options can be specified as needed:

Permission	Description
all	All permissions
annotate	User can annotate document
copy	User can copy text and images from document
modify	User can modify document
print	User can print document
no-annotate	User cannot annotate document
no-copy	User cannot copy text and images from document
no-modify	User cannot modify document
no-print	User cannot print document
none	No permissions

This option is only available when generating PDF files.

--portrait

The `--portrait` option specifies that the output should be in portrait orientation (short edge on top).

This option is only available when generating PostScript or PDF files.

--pscommands

The `--pscommands` option specifies that PostScript device commands should be written to the output files.

This option is only available when generating Level 2 and Level 3 PostScript files.

--quiet

The `--quiet` option prevents error messages from being sent to stderr.

--right margin

The `--right` option specifies the right margin. The default units are points (1 point = 1/72nd inch); the suffixes "in", "cm", and "mm" specify inches, centimeters, and millimeters, respectively.

This option is only available when generating PostScript or PDF files.

--size size

The `--size` option specifies the page size. The `size` parameter can be one of the following standard sizes:

size	Description
Letter	8.5x11in (216x279mm)
A4	8.27x11.69in (210x297mm)
Universal	8.27x11in (210x279mm)

Custom sizes are specified by the page width and length separated by the letter "x" to select a custom page size. Append the letters "in" for inches, "mm" for millimeters, or "cm" for centimeters.

This option is only available when generating PostScript or PDF files. Use the `--pscommands` option to generate PostScript page size commands.

--strict

The `--strict` option turns on strict HTML conformance checking. When enabled, HTML elements that are improperly nested and dangling close elements will produce error messages.

--textcolor color

The `--textcolor` option specifies the default text color for all pages in the document. The color can be specified by a standard HTML color name or as a 6−digit hexadecimal number of the form `#RRGGBB`.

--textfont typeface

The `--textfont` options sets the typeface that is used for text in the document. The `typeface` parameter can be one of the following:

typeface	Actual Font
Arial	Helvetica
Courier	Courier
Helvetica	Helvetica
Monospace	Courier
Sans−Serif	Helvetica
Serif	Times
Symbol	Symbol
Times	Times

--title

The `--title` option specifies that a title page should be generated.

--titlefile filename

The `--titlefile` option specifies a HTML file to use for the title page.

--titleimage filename

The `--titleimage` option specifies the title image for the title page. The supported formats are BMP, GIF, JPEG, and PNG.

--tocfooter lcr

The `--tocfooter` option specifies the contents of the table-of-contents footer. The `lcr` parameter is a three-character string representing the left, center, and right footer fields. See the `--footer` option for the list of formatting characters.

Setting the TOC footer to "`...`" disables the TOC footer entirely.

--tocheader lcr

The `--tocheader` option specifies the contents of the table-of-contents header. The `lcr` parameter is a three-character string representing the left, center, and right header fields. See the `--footer` option for the list of formatting characters.

Setting the TOC header to "`...`" disables the TOC header entirely.

--toclevels levels

The `--toclevels` options specifies the number of heading levels to include in the table-of-contents pages. The `levels` parameter is a number from 1 to 6.

--toctitle string

The `--toctitle` options specifies the string to display at the top of the table-of-contents; the default string is "Table of Contents".

--top margin

The `--top` option specifies the top margin. The default units are points (1 point = 1/72nd inch); the suffixes "in", "cm", and "mm" specify inches, centimeters, and millimeters, respectively.

This option is only available when generating PostScript or PDF files.

--user-password password

The `--user-password` option specifies the user password for a PDF file. If not specified or the empty string (""), no password will be required to view the document.

This option is only available when generating PDF files.

--verbose

The `--verbose` option specifies that progress information should be sent/displayed to the standard error file.

--version

The `--version` option displays the HTMLDOC version number.

--webpage

The `--webpage` option specifies that the input files comprise a web page (or site) and that no title page or table-of-contents should be generated. *HTMLDOC* will insert a page break between each input file.

This option is only available when generating PostScript or PDF files.

--xrxcomments

The `--xrxcomments` option specifies that Xerox PostScript job comments should be written to the output files.

This option is only available when generating PostScript files.

Messages

HTMLDOC sends error and status messages to stderr unless the `--quiet` option is provided on the command-line. Applications can capture these messages to relay errors or statistics to the user.

BYTES: Message

The `BYTES:` message specifies the number of bytes that were written to an output file. If the output is directed at a directory then multiple `BYTES:` messages will be sent.

PAGES: Message

The `PAGES:` message specifies the number of pages that were written to an output file. If the output is directed at a directory then multiple `PAGES:` messages will be sent. No `PAGES:` messages are sent when generating HTML output.

ERRnnn: Messages

The `ERRnnn:` messages specify an error condition. Error numbers 1 to 13 map to the following errors:

1. No files were found or loadable.
2. No pages were generated.
3. The document contains too many files or chapters.
4. *HTMLDOC* ran out of memory.
5. The specified file could not be found.
6. The comment contains a bad *HTMLDOC* formatting command.
7. The image file is not in a known format.
8. *HTMLDOC* was unable to remove a temporary file.
9. *HTMLDOC* had an unspecified internal error.
10. *HTMLDOC* encountered a networking error when retrieving a file via a URL.
11. *HTMLDOC* was unable to read a file.
12. *HTMLDOC* was unable to write a file.
13. A HTML error was found in a source file.

Error numbers 100 to 505 correspond directly to a HTTP status code.

Appendix A − GNU General Public License

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

> a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
>
> b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
>
> c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

> a. Accompany it with the complete corresponding machine−readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
>
> b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine−readable

copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty−free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made

generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix B − Book File Format

This appendix describes the *HTMLDOC .book* file format.

Introduction

The *HTMLDOC .book* file format is a simple text format that provides the command−line options and files that are part of the document. These files can be used from the GUI interface or from the command−line using the `−−batch` option:

```
htmldoc filename.book
htmldoc −−batch filename.book
```

The first form will load the book and display the GUI interface, if configured. Windows users should use *ghtmldoc.exe* executable to show the GUI and *htmldoc.exe* for the batch mode:

```
ghtmldoc.exe filename.book
htmldoc.exe −−batch filename.book
```

The Header

Each *.book* file starts with a line reading:

```
#HTMLDOC 1.8.17
```

The version number (1.8.17) is optional.

The Options

Following the header is a line containing the options for the book. You can use any valid command−line option on this line:

```
−f htmldoc.pdf −−titleimage htmldoc.png −−duplex −−compression=9 −−jpeg=90
```

Long option lines can be broken using a trailing backslash (\) on the end of each continuation line:

```
−f htmldoc.pdf −−titleimage htmldoc.png −−duplex \
−−compression=9 −−jpeg=90
```

The Files

Following the options are a list of files or URLs to include in the document:

```
intro.html
1−install.html
2−starting.html
3−books.html
4−cmdline.html
5−cgi.html
6−htmlref.html
7−guiref.html
8−cmdref.html
a−license.html
b−book.html
c−relnotes.html
```

Putting It All Together

The following is the complete book file needed to generate this documentation:

```
#HTMLDOC 1.8.13
−f htmldoc.pdf −−titleimage htmldoc.png −−duplex −−compression=9 −−jpeg=90
intro.html
1−install.html
2−starting.html
3−books.html
4−cmdline.html
5−cgi.html
6−htmlref.html
7−guiref.html
8−cmdref.html
a−license.html
b−book.html
c−relnotes.html
```

Older Book Files

Prior to *HTMLDOC* version 1.8.12, the book file format was slightly different:

```
#HTMLDOC version
file count
file(s)
options
```

While *HTMLDOC* still supports reading this format, we do not recommend using it for new books. In particular, when generating a document using the `--batch` option, some options may not be applied correctly since the files are loaded prior to setting the output options in the old format.

Appendix C − Release Notes

This appendix provides the release notes for each version of *HTMLDOC*.

Changes in HTMLDOC v1.8.20

New Features

- New −−nup and NUMBER−UP options for PostScript and PDF output.
- HTMLDOC now logs HTML errors.
- HTMLDOC now supports the A3, B, Legal, and Tabloid size names.
- HTMLDOC now supports embedding of the base Type1 fonts in PostScript and PDF output.

Changes

- The HTML parser now allows BODY to auto−close HEAD and visa−versa.

Bug Fixes

- HTMLDOC wouldn't compile using GCC under HP−UX due to a badly "fixed" system header file (vmtypes.h).
- Generating a book without a table−of−contents would produce a bad PDF file.
- The Xerox XRX comments used the wrong units for the media size, points instead of millimeters.
- IMG elements with links that use the ALIGN attribute didn't get the links.
- Header and footer comments would interfere with the top and bottom margin settings.
- Fixed a bug in the htmlReadFile() function which caused user−provided title pages not to be displayed in PS or PDF output.

- The table−of−contents would inherit the last media settings in the document, but use the initial settings when formatting.

Changes in HTMLDOC v1.8.19

New Features

- Now support the "subject" meta variable.

Changes

- Updated the HTML parser to use HTML 4.0 rules for embedding elements inside a LI.
- Now check for a TYPE attribute on EMBED elements, so that embedded Flash files do not get treated as HTML.
- Now put the COPYRIGHT meta data in the Author field in a PDF file along with the AUTHOR meta data (if any).
- No longer embed the prolog.ps command header when PostScript commands are not being embedded in the output.
- *HTMLDOC* now properly ignores the HTML 4.0 COL element.

Bug Fixes

- Squeezed tables were not centered or right−aligned properly.
- Cells didn't align properly if they were the first things on the page, or if there were several intervening empty cells.
- The preferred cell width handling didn't account for the minimum cell width, which could cause some tables to become too large.
- Remote URLs didn't always resolve properly (like the images from the Google web page...)
- The font width loading code didn't force the non−breaking space to have the same width as a regular space.
- PRE text didn't adjust the line height for the tallest fragment in the line.
- *HTMLDOC* tried to seek backwards when reading HTML from the standard input.
- The media margin comments did not work properly when the current media orientation was landscape.

Changes in HTMLDOC v1.8.18

New Features

- Added support for remote HTML title pages.

Changes

- Now accept all JPEG files, even if they don't start with an APPn marker.
- Now only start a new page for a chapter/filter if we aren't already at the top of a page.

Bug Fixes

- ROWSPAN handling in tables has been updated to match the MSIE behavior, where the current rowspan is reduced by the minimum rowspan in the table; that is, if you use "ROWSPAN=17" for all cells in a row, *HTMLDOC* now treats this as if you did not use ROWSPAN at all. It is unclear if this is what the W3C intends.
- The "−−webpage" option didn't force toc levels to 0, which caused a bad page object reference to be inserted in the PDF output file.
- Background colors in nested tables didn't always get drawn in the right order, resulting in the wrong colors showing through.
- The HEADER page comment didn't set the correct top position in landscape orientation.

Changes in HTMLDOC v1.8.17

New Features

- Improved table−of−contents generation, with chapter headings at the top of new TOC pages and page numbers based on the header/footer string.
- Added new "−−no−localfiles" option to disable access to local files for added security in web services.
- Long lines in book files can not be broken up using a trailing backslash.
- Added a modern "skin" to the GUI interface.

Changes

- Made some changes in how COLSPAN and ROWSPAN are handled to better match how Netscape and MSIE format things.
- *HTMLDOC* now handles .book files with CR, LF, or CR LF line endings.
- Changed the TOC numbering to use 32−bit integers instead of 8−bit integers...
- Now handle local links with quoted (%HH) characters.
- The command−line interface no longer sets PDF output mode when using −−continuous or −−webpage.
- *HTMLDOC* now opens HTML output files in binary mode to prevent extra CR's under Windows, and strips incoming CR's from PRE text.
- Now support inserting the current chapter and heading in the table−of−contents headers and footers.

Bug Fixes

- The table cell border and background were offset by the cellpadding when they should only be offset by the cellspacing.
- The buffer used for periods that lead up to the page number in the table−of−contents was not large enough for a legal−size document in landscape format.
- If a book only contained chapter headings, the PDF bookmarks would be missing the last chapter heading.
- Table cells that ended with a break would render incorrectly.
- Fixed the table pre−format sizing code to properly account for borders, padding, etc.
- Fixed the table squeezing code to honor minimum widths and properly resize the remaining space.
- The MEDIA SIZE page comment did not reset the printable width and length of the page.
- Tables that used COLSPAN did not honor WIDTH values in non−spanned cells.

Changes in HTMLDOC v1.8.16

Changes

- Now break before and after DIV groups to match most browsers (the HTML spec is ambivalent about it...)

Bug Fixes

- HR elements didn't render properly.
- Background images didn't render properly and could lock up *HTMLDOC*.
- The "HALF PAGE" comment would lock up *HTMLDOC* − *HTMLDOC* would keep adding pages until it ran out of memory.
- SUP and SUB used a fixed (reduced) size instead of using a smaller size from the current one.
- Empty cells could cause unnecessary vertical alignment on the same row.

Changes in HTMLDOC v1.8.15

New Features

- Now support media source, type, and color attributes in PS output.
- Now support per−page size, margins, headers, footers, orientation, and duplexing.
- Now support plain text for headers and footers, with $ variables to include page numbers and so forth.
- New device control prolog file for printer−specific option commands.
- Now support a new continuous web page mode that doesn't automatically insert a page break with each HTML file or URL (−−continuous).
- Now draw border around inline images as needed.
- Now support MacOS X (only command−line at present).
- Now support the "page−break−before", "text−align", "vertical−align" style attributes, but only for style information in an element's STYLE attribute.

Changes

- Now load images into memory only as needed, and unload them when no longer needed. This provides a dramatic reduction in memory usage with files that contain a lot of in−line images.
- Now use the long names for the Flate and DCT filters in all non−inline PDF streams. This avoids a stupid bug in Acrobat Reader when printing to PostScript printers.
- *HTMLDOC* now strips any trailing GET query information when saving the start of files (target) in a document.
- Unqualified URLs (no leading scheme name, e.g. http:) now default to the HTTP port (80) instead of the IPP port (631).
- Optimized the image writing code to do more efficient color searching. This provides a significant speed improvement when including images.
- Now hide all text inside SCRIPT, SELECT, and TEXTAREA elements.
- OS/2 port changes from Alexander Mai.

Bug Fixes

- If a document started with a heading greater than H1, *HTMLDOC* would crash.
- Full justification would incorrectly be applied to text ending with a break.
- Images using ALIGN="MIDDLE" were not centered properly on the baseline.
- Table cells that used both ROWSPAN and COLSPAN did not format properly (the colspan was lost after the first row.)
- Tables that used cells that exclusively used COLSPAN did not format properly.
- When writing HTML output, image references would incorrectly be mapped using the current path.
- Images with a width or height of 0 should not be written to PS or PDF output.
- The CreationDate comment in PostScript output contained a bad timezone offset (+−0500, for example, instead of −0500).
- The PHP portal example now verifies that the URL passed to it contains no illegal characters.

Changes in HTMLDOC v1.8.14

New Features

- Added support for 128−bit encryption.
- Added support for GET form request data in the PHP and Java "portal" examples.

Changes

- Most output generation limits have been removed; *HTMLDOC* now dynamically allocates memory as needed for pages, images, headings, and links. This has the happy side−effect of reducing the initial memory footprint significantly.
- Now call setlocale() when it is available to localize the date and time in the output.
- The table parsing code now checks to see that a ROWSPAN attribute fits in the table; e.g., a ROWSPAN of 10 for a table that has only 6 rows remaining needs to be reduced to 6...

Bug Fixes

- Tables with a lot of COLSPANs could cause a divide− by−zero error or bad pages (NAN instead of a number.)
- Table cells with a single render element would not be vertically aligned.
- The −−quiet option would enable progress messages on the command−line.
- Table cell widths could be computed incorrectly, causing unnecessary wrapping.

Changes in HTMLDOC v1.8.13

New Features

- Added support for secure (https) URLs via the OpenSSL library.
- Added support for Acrobat 5.0 (PDF 1.4).
- Added support for transparency in PostScript and PDF 1.1 and 1.2 output.
- Added a −−no−jpeg option (same as −−jpeg=0)
- Added support for the CSS2 page−break−before and page−break−after properties.
- Added a PHP example.

Changes

- External file references to non−PDF files now use the "Launch" action so they can be opened/executed/saved as allowed by the OS and PDF viewer.
- Changed the indexed/JPEG'd transition point to 256 colors when using Flate compression. This makes PDF files much smaller in general.
- Changed the in−line image size limit to 64k.
- Now allow "<" followed by whitespace, "=", or "<". This violates the HTML specification, but we're sick of people complaining about it.
- Preferences are now stored in a user−specific file under Windows, just like UNIX. This provides user−specific preferences and allows preferences to be kept when upgrading to new versions of *HTMLDOC*.
- The book loading code now allows for blank lines, even though these are not a part of the format. (added to support some scripted apps that include extra newlines...)
- Changed the leading space handling of blocks to more closely match the standard browser behavior.

Bug Fixes

- The table formatting code adding the border width to the cell width, while Netscape and MSIE don't. This caused some interesting formatting glitches...
- The table formatting code didn't account for the preferred width of colspan'd cells.
- The table formatting code tried to enforce the minimum cell width when squeezing a table to fit on the page; this caused the table to still exceed the width of the page.
- The PDF catalog object could contain a reference to a /Names object of "0 0 R", which is invalid. This would happen when the "−−no−links" option was used.
- Several HTML elements were incorrectly written with closing tags.
- When piping PDF output, the temporary file that is created needed to be open for reading and writing, but *HTMLDOC* only opened the file for writing.
- Image links did not work.
- The JPEG image loading code did not correctly handle grayscale JPEG images.
- JPEG images were not encrypted when writing a document with encryption enabled.
- The user password was not properly encrypted.
- The colormap of indexed images were not encrypted when writing a document with encryption enabled.
- The temporary file creation and cleanup functions did not use the same template under Windows, causing multiple conversions to fail when temporary files were used.
- Paragraphs could end up with one extra text fragment, causing the line to be too long.
- The command−line program would clear the error count after reading all the files/URLs on the command−line, but before generating the document. If there were problems reading the files/URLs, *HTMLDOC* would return a 0 exit status instead of 1.
- Image objects that were both JPEG and Flate compressed would not display (filters specified in the wrong order.)
- Images with more than 256 colors would cause a segfault on some systems.
- Background images would generate the error message "XObject 'Innn' is unknown".

Changes in HTMLDOC v1.8.12

New Features

- Added new "−−batch" option to convert *HTMLDOC* book files from the command−line.
- Added support for the "−display" option on systems that use X11.
- Now use image objects in PDF output for images when the image width * height * depth > 32k.
- Now use JPEG compression when the number of colors would be > 32 colors or 16 gray shades.
- True transparency support for GIF files in PDF 1.3 output!
- The GUI now automatically changes the extension of the output filename as needed.
- The GUI now collects all error messages and shows them once after the document is generated.
- Added support for HSPACE and VSPACE attributes for images with ALIGN="LEFT" or ALIGN="RIGHT".
- Added new Java interface to *HTMLDOC*.

Changes

- Consolidated temporary file management into new file_temp() function. The new function also makes use of the Windows "short lived" open option which may improve performance with small temporary files.
- Updated book file format and added an appendix describing the format.
- Now default to PDF 1.3 (Acrobat 4.0) output format.
- Now output length of PDF streams with the stream object; this offers a modest reduction in file size.
- The HTTP file cache now keeps track of previous URLs that were downloaded.
- The HTTP code now supports redirections (status codes 301 to 303) to alternate URLs.
- Limit the height check for table rows to 1/8th of the page length; this seems to provide fairly consistent wrapping of tables without leaving huge expanses of blank space at the bottom of pages.
- The HTML output now also includes a font−family style for PRE text; otherwise the body font would override the PRE font with some browsers.
- The snprintf/vsnprintf emulation functions were not included in the *HTMLDOC* makefile.
- RGB hex colors are now recognized with or without the leading #. This breaks HTML standards compliance but should reduce the number of problem reports from buggy HTML.
- The stylesheet generated with the HTML output no longer contains absolute font sizes, just the typefaces and a relative size for SUB/SUP.
- The title image is no longer scaled to 100% in the HTML output.

Bug Fixes

- The web page output was not divided into chapters for each input file.
- The "make install" target did a clean.
- The configure script would remove the image libraries if you did not have FLTK installed.
- The fix_filename() function didn't handle relative URLs for images (e.g. SRC="../images/filename.gif")
- Comments in the source document were being closed by a ".
- The command−line and GUI interfaces looked for "outlines" instead of "outline" for the page mode.
- The HTML output code didn't output closing tags for empty elements.
- The GUI interface started with the compression slider enabled, even for HTML output.
- The beginnings of some lines could start with whitespace.
- Wasn't aligning images and text on lines based on the line height.
- The compression slider was enabled in the GUI even though HTML output was selected.
- The Perl example code was incorrect.
- Fixed the check for whether or not pages were generated.

- htmlSetCharSet() wasn't reloading the character set data if the data directory changed.
- The GUI did not reset the default background color.
- The 'C' page number style (chapter page numbers) started at 3 instead of 1.
- The chapter links were off by 1 or 2 pages when no title page was included.

Changes in HTMLDOC v1.8.8

New Features

- Added support for PDF security/encryption!
- Now support TABLE height attribute.
- Now generate an error message if no pages are generated (with a suggestion to use the webpage option.)
- New "paths" option to specify additional directories to search for files. This is useful when the source files use absolute server paths.

Changes

- Added missing casts in htmllib.cxx that were causing a compile warning with some compilers.
- No longer draw borders around empty cells in tables...
- Now disable the TOC tab when using webpage mode.
- Now scale title image to 100% in HTML output.
- Now handle comments with missing whitespace after the "<!−−".

Bug Fixes

- Nested tables didn't take into account the table border width, spacing, or padding values.
- *HTMLDOC* crashed under Solaris when reading HTML files from the standard input.
- <ELEM>text</ELEM> <MELE>text</MELE> was rendered without an intervening space.

Changes in HTMLDOC v1.8.7

New Features

- The configure script now uses the local PNG, ZLIB, and/or JPEG libraries when they are new enough.
- The configure script now uses the −fno−rtti, −fno−exceptions, and −fpermissive options as needed with GCC (smaller, faster executables, works around X header bugs in Solaris.)
- Added a −−toctitle option to set the table−of−contents title from the command−line (was only available in the GUI in previous releases...)
- New "<!−− NEED amount −−>" comment to force a page feed if there is not sufficient room on the page for the following text.
- Page comments are now supported in tables.
- Table rows are now allocated dynamically, MAX_ROWS at a time.

Changes

- Increased default MAX_PAGES to 10000 (was 5000.)
- File links in book files now point to the top of the next page.

- <TABLE ALIGN=xyz> now aligns the table (previously it just set the default alignment of cells.)
- Transparent GIFs now use the body color instead of white for the transparent color.
- Updated to LIBPNG 1.0.6 in source distribution.
- Updated the default cellpadding to be 1 pixel to match Netscape output.
- Updated line and block spacing to match Netscape.
- DL/DT/DD output now matches browsers (was indented from browser output.)
- Now only output link (A) style if it is set to "none". Otherwise Netscape would underline all targets as well as links.
- Increased the MAX_COLUMNS constant to 200, and dropped MAX_ROWS to 200. Note that the new table code now allocates rows in increments of MAX_ROWS rows, so the actual maximum number of rows depends on available memory.

Bug Fixes

- Now ignore illegal HTML in tables.
- The VALIGN code didn't handle empty cells properly.
- Wasn't offsetting the start of each row by the cell padding.
- The JPEG image loading code didn't work for some JPEG images, particularly those from digital cameras (JPEG but not JFIF format.)
- The strikethrough line was not being drawn in the correct position.
- Wasn't setting the height of BR elements, so

 didn't insert a blank line.
- The table of contents would show the wrong page numbers if no title page was generated.
- Cell widths did not subtract any border, padding, or spacing from the "preferred" width, causing formatting differences between web browsers and *HTMLDOC*.
- The PNG loading code did not handle interlacing or transparency.
- The HTML parsing code did not prevent elements in embedded files from completing elements in the parent file.
- The table CELLSPACING amount was being applied twice in the table sizing calculations.

Changes in HTMLDOC v1.8.6

New Features

- New linkcolor and linkstyle options.

Changes

- Minor source changes for OS/2 compilation.
- SUP and SUB now raise/lower text more to be consistent with browser look−n−feel.
- Non−breaking space by itself was being output. Now check for that and ignore strings that consist entirely of whitespace.
- New progress bar.

Bug Fixes

- Didn't add whitespace after a table caption.
- Nested tables caused formatting problems (flatten_tree() didn't insert breaks for new rows)
- A cell whose minimum width exceeded the available width for the table would cause the table to go off the page.

- Cells that spanned more than two pages were drawn with boxes around them rather than just the sides.
- The stylesheet info in the HTML output specified the H1 size for all headings.
- The title page was incorrectly formatted when an image was specified − the text start position was computed using the pixel height of the title image and not the formatted height.
- 1 color images didn't come out right; the "fix" to work around an Acrobat Reader bug was being done too soon, so the color lookups were wrong.
- HTML file links now work properly.
- Now limit all HTML input to the maximum size of input buffers to avoid potential buffer overflow problems in CGIs.
- If a row had a predefined height, *HTMLDOC* wasn't making sure that the row would fit on the current page.
- THEAD, TFOOT, and TBODY caused problems when formatting tables. Note: THEAD and TFOOT are *still* not supported, however the code now properly ignores them and parses the rows in the TBODY group.
- The VALIGN code introduced in the 1.8.5 release didn't check for NULL pointers in all cases.

Changes in HTMLDOC v1.8.5

New Features

- New "−−titlefile" option to include an HTML file for the title page(s).
- New 'C' header/footer option to show current page number within chapter or HTML file.
- Allow adding of .book files to import all HTML files in the book.
- New "HALF PAGE" page comment to feed 1/2 page.
- Added VALIGN and HEIGHT support in tables.

Changes

- Now optimize link objects in PDF files (provides a 40k reduction in file size for the *HTMLDOC* manual alone)
- Table rows that cross page boundaries are now rendered more like Netscape and MSIE.
- Now support HTMLDOC_DATA and HTMLDOC_HELP environment variables under UNIX (for alternate install directory)
- Now show error messages when *HTMLDOC* can't open the AFM, character set, or PostScript glyph files.
- The logo image is now scaled to its "natural" size (as it would appear in a web browser)
- Now recognize VALIGN="MIDDLE" or VALIGN="CENTER".

Bug Fixes

- Generation of PDF files to the standard output (i.e. to the web server + browser) didn't work on some versions of UNIX. *HTMLDOC* now writes the PDF output to a temporary file and then copies it to the standard output as needed.
- PDF links were missing the first 5 characters in the filename; the code was trying to skip over the "file:" prefix, but that prefix was already skipped elsewhere.
- Nested descriptive lists (DL) did not get rendered properly.
- Tables had extra whitespace before and after them.
- Multiple aligned images confused parse_paragraph(); the images would overlap instead of stack on the sides.

Changes in HTMLDOC v1.8.4

Changes

- More configure script changes for FLTK DSOs.
- FileIcon.cxx was still using NULL for outline (an integer), which caused some ANSI C++ compilers to complain.

Bug Fixes

- The Fonts and Colors tab groups did not extend to the full width of the tab area, which prevented the Browse button from working when clicked on the right side.
- The help dialog window did not scroll all the way to the bottom of the text.
- The chapter ("c") header/footer string did not work.
- The heading ("h") header/footer string did not always match the first heading on a page.
- The header and footer fonts were not used when computing the widths of the header and footer strings.
- The Windows distribution did not create the right shortcut for the Users Manual in the Start menu.
- The command−line code did not accept "−−grayscale", only "−−gray"
- Multi−file HTML output did not use the right link for the table−of−contents file if no title page was being generated.
- Extra whitespace before and after tables has been eliminated.

Changes in HTMLDOC v1.8.3

New Features

- New "−−browserwidth" option to control scaling of images and tables that use fixed pixel widths.

Changes

- The configure script now looks for the OpenGL library (required if you use a shared FLTK library with OpenGL support.)
- Increased the max number of chapters to 1000.

Bug Fixes

- Page break comments didn't force a paragraph break.
- −−no−toc prevented chapters from being output in PS and PDF files.
- Filenames didn't always get updated properly when doing a "save as"...
- Fixed some more leading/trailing whitespace problems.
- Wasn't freeing page headings after the document was generated.
- Wasn't range checking the current chapter number; now limits the number of chapters to MAX_CHAPTERS and issues an error message whenever the limit is exceeded.

Changes in HTMLDOC v1.8.2

New Features

- New "setup" program for UNIX software installation.

Changes

- Documentation updated for new UNIX "setup" program and "..." usage for headers and footers.
- Changed margins to floating point (instead of integer) to improve table column accuracy.

Bug Fixes

- *HTMLDOC* could crash under Microsoft Windows with some types of HTML files. This was caused by a stack overflow, usually when processing nested tables.
- Multiple HTML files weren't being converted properly in web page mode − only the last file would be generated for PostScript output, and no file for PDF output.
- Wasn't preserving the whitespace between "one" and "two" in the HTML code "one<I> two</I> three".
- Paragraph spacing was inconsistent.
- <TABLE WIDTH="xx"> wasn't formatted properly.
- The command−line code wasn't opening HTML files in binary mode. This caused problems under Microsoft Windows.

Changes in HTMLDOC v1.8.1

Changes

- The configure script didn't update the ARFLAGS variable for *BSD operating systems (no "s" option to build the symbol table...)
- Changed the installation commands to only create the installation directory if it does not exist. This prevents installation errors on some platforms the second time around.
- Now use the Microsoft definitions for characters 128 through 159 that are otherwise unused by the ISO−8859−x character sets.
- Now set optimization settings when we know the compiler.
- Now always quote attribute values in HTML output to make HTML lint programs happy.

Bug Fixes

- Wasn't using TOC title string in PDF document outline.
- Preformatted text in tables didn't force the column width.
- Cells using COLSPAN > 1 didn't contribute to the width of columns.
- The table code didn't enforce the per−column minimums under certain circumstances, causing "scrambled" columns.
- The configure script and makefiles didn't work when FLTK was not available. They now only build the "gui" library when it is available.
- The Windows distribution was installing files under PROGRAMDIR instead of TARGETDIR. This prevented users from customizing the installation directory.
- The configure script overrode the LDFLAGS environment variable, preventing FLTK from being located in a non− default directory.

Changes in HTMLDOC v1.8

New Features

- Now support PDF 1.1 (Acrobat 2.x) and PDF 1.3 (Acrobat 4.0).
- Now support PDF page modes, layouts, and effects, and the first page that is displayed in Acrobat Reader.
- Now support PostScript Level 3 output with Flate image compression.
- Now support PostScript commands for page size and duplexing.
- Now add filenames as needed to HTML links.
- Added optimizations to output code to further reduce PDF and PostScript file size.
- Now support alternate 8−bit character sets. Currently we supply data files for the ISO−8859−N character sets.
- Added chapter headings to the available header/footer formats.
- The GUI file chooser is significantly improved and supports selection of multiple HTML files.
- The GUI now provides on−line help.
- Many other GUI improvements.
- Added support for DIR and MENU block elements.
- The header and footer text can now be made boldface, italic, etc.
- Font settings are now exported to HTML files in a style sheet.
- Now support page breaks using HTML comments.
- The image dimensions are now exported to HTML files.
- Added landscape printing option.
- Added CAPTION support for tables.
- Filename links now work for HTML files included in a document.
- Now support BGCOLOR in tables.

Changes

- Lots of documentation changes.
- Much better table formatting.
- Changed HTML output to use less invasive navigation bars at the top and bottom of each file. This also means that the "−−barcolor" option is no longer supported!
- Updated to use existing filenames in HTML (directory) output.
- Now recognize any local PDF file as a local file link (i.e. you just need "HREF=filename.pdf" and not "HREF=file:filename.pdf")
- <TT>, <CODE>, and <SAMP> no longer reduce the font size.
- Now put whitespace after image data in PDF files. This change was needed to work around a bug in Acrobat Reader 4.0.
- Now generate a complete encoding vector for fonts in PDF files. This change was needed to work around a bug in all versions of Acrobat Exchange that did not recognize the WinANSI encoding defined in the PDF specifications.
- Now filter out the BREAK attribute from HR elements.
- Now only load images once.

Bug Fixes

- Wasn't escaping &,<, or > in HTML output
- Wasn't preserving
- Links in multi−file HTML output were off−by−one.

- BLOCKQUOTE needed to be like CENTER and DIV.
- Needed to use existing link name if present for headings to avoid nested link name bug in Netscape and MSIE.
- Extremely long link names could cause TOC generation to fail and *HTMLDOC* to crash.
- PDF output was not compatible with Ghostscript/Ghostview because Ghostscript does not support inherited page resources or the "Fl" abbreviation for the "FlateDecode" compression filter.
- PostScript DSC comments didn't have unique page numbers. This caused Ghostview (among others) to get confused.
- Some functions didn't handle empty text fragments.
- Images couldn't be scaled both horizontally and vertically.
- <LI> didn't support the VALUE attribute (but <OL> did...)
- Fixed whitespace problems before and after some markups that was caused by intervening links.
- The indexed image output code could generate an image with only 1 color index used, which upset Acrobat Reader.
- Fixed a bug in table−of−contents handling − *HTMLDOC* would crash on some systems if you converted a web page on the command−line.
- Wasn't setting the font size and spacing soon enough when generating files on the command−line.
- Didn't hide EMBED elements when generating indexed HTML files.
- Didn't always set the current drawing position before drawing a box or line.
- Base85 encoding of image data was broken for PostScript output.
- JPEG compression was broken for PostScript output.
- Didn't set binary mode for the standard output under Windows and OS/2 needed.