Tidydoc Manual

The Documentation Organizer
Edition 0.4, for Tidydoc Version 0.4
3 January 2009

by Nicolas Burrus

This file documents the `tidydoc` command for generating a documentation browser.

Short Contents

Table of Contents

1 Overview

1.1 License

This program is licensed under the Gnu GPL license http://www.gnu.org/copyleft/gpl.html.

1.2 Introduction

The goal of the `tidydoc` program is to make documentation organization easier. It addresses the following problems:

- Add documents to a pool of documents
- Give them the right names and put them into the right directories
- Generate a set of HTML pages to browse your documents

`tidydoc` is written in Python in a spirit of flexibility and easy tweaking, but with default behaviors which fit the needs of most people.

You can add documents using arbitrary commands, such as a simple `cp` or through ssh with `scp`. There is also a web interface for those running a webserver.

`tidydoc` can then generate HTML files indexing your documents. An XML description file is required for each document. These XML files can be generated from very simple text files (`.dsc` files).

No additional tool is required, a standard Unix system with a Python interpreter will do the job.

1.3 Features

Version 0.4 includes the following features:

- Documents uploading
  - Simple pre-filled text files to describe documents
  - Any upload command can be used, or a web interface
  - Multiple categories are supported
  - Files are automatically renamed using title, author and date information
  - Files are automatically put into the right directories
- HTML generation
  - Handy navigation panel using Treeview http://www.treeview.net
  - Some directories may be considered as row documentation
  - Flexible output using templates
  - Descriptions can be edited from a web interface
  - Support search queries
  - New default template with folding support
- Bibtex generation

`tidydoc` was written by Nicolas Burrus nicolas.burrus@ensta.fr.

Many concepts were inspired from Orgadoc http://www.gnu.org/software/orgadoc/.

1.4 Quick start for a local filesystem

This section describes the simplest way to start using `tidydoc`. This will be done through a commented sequence of shell commands:

```
$ tar xvfj tidydoc-0.4.tar.bz2 # uncompress the sources
$ cd tidydoc-0.4
$ ./configure --prefix=/usr/local # files will get installed in /usr/local
$ make # generate files and documentation
$ make install # installation, you might need to be root
$ cd
$ mkdir docs
$ td-create-htmlroot docs # initialize the documentation directory
$ emacs config/tidydoc.conf # default parameters should be ok
```

Now you can start adding documents:

```
$ td-add-doc -c ~/docs/config/tidydoc.conf my_favorite_doc.pdf
$ td-add-doc -c ~/docs/config/tidydoc.conf \
             http://nicolas.burrus.name/tidydoc_manual.pdf
```

Or if you use only one document pool, you can copy the configuration file to your home directory, and it will be used by default (the `-c` option will not be required anymore):

```
$ cp ~/docs/config/tidydoc.conf ~/.tidydoc.conf
$ td-add-doc http://nicolas.burrus.name/tidydoc_manual.pdf
```

By default `td-add-doc` will open `emacs` to edit the description file, and `acroread` to visualize the document (so that you can easily copy/paste elements of the pdf to fill in the description). This can be changed at the end of the configuration file.

Once the description file has been filled, quit `emacs` and `acroread`. If you did not enter any category, the document will be put in "unsorted".

Then you can generate html and bibtex files:

```
$ td-generate -c ~/docs/config/tidydoc.conf
  => ~/docs/index.html gives access to your documentation tree
  => ~/docs/documents.bib now contains the bibtex entries
```

1.5 Quick start with a web server

This is very similar to the previous example, but documents will be managed through the web interface.

```
$ tar xvfj tidydoc-0.4.tar.bz2 # uncompress the sources
$ cd tidydoc-0.4
$ ./configure --prefix=/usr/local # files will get installed in /usr/local
$ make # generate files and documentation
$ make install # installation, you might need to be root
$ cd
$ mkdir /var/www/docs
$ td-create-htmlroot /var/www/docs # initialize the documentation directory
$ emacs /var/www/docs/config/tidydoc.conf
```

Here you must change the `site_root` variable to your website address:

```
site_root = "http://my.website.com/doc"
```

Now you have to configure your web server. Here is an example for apache:

```
Alias /doc /var/www/docs
<Directory "/var/www/docs">
           Options FollowSymLinks
           AllowOverride None
</Directory>

<Directory "/var/www/docs/cgi-bin">
           AllowOverride None
           AddHandler cgi-script .cgi
           Options ExecCGI -MultiViews +SymLinksIfOwnerMatch
           Order allow,deny
           Allow from all
</Directory>
```

Now generate the initial (empty) website:

```
$ td-generate -c /var/www/docs/config/tidydoc.conf
```

Opening your favorite browser to http://my.website.com/doc you should be able to upload, modify and browse your documents.

2 Adding documents

2.1 Using `td-add-doc`

To add documents, use the `td-add-doc` command.

The format for running the `td-add-doc` program is:

```
Usage: td-add-doc [options] document_file [description_file]

  Options

  -c, --config configfile  Config file to use.
  -v, --version            Print version number.
```

'`document_file`'
: Document to add. The name of the document does not matter.

'`description_file`'
: Template to use to create the `.dsc` file. If not specified, `tidydoc` will generate one automatically.

`td-add-doc` open the document and ask you to fill out a `.dsc` file describing its content. A confirmation is required before actually sending the file.

Destination directories and file names will be determined from the description file. If a new category is introduced, a new directory will be automatically created. The naming scheme for files is '`first_author.year.conf.title.extension`'.

2.2 `.dsc` file format

`.dsc` files are simples. Here is a sample file:

```
% This is a comment
Title: Fractals are everywhere in real world

% One author per line
Author: Patrick Foobar

Date: 1989

Conf: micai

Pages: 4

Keywords: fractals nature

% Url towards the document.
Link: http://patrick.foobar.com/fractals.pdf

% Line breaks are taken into account.
Abstract: Fractional Brownian motions number of physical phenomena
```

```
fractional order. However, the precise meaning of such complementary
approaches are The first one, based on nonstationary nature of
measurements; the second one, self-similarity properties of FBM and
reveals an underlying stationary structure relative to each
time-scaling.

% public or private
Visibility: public

Language: english

% Kind of bibtex entry
Bibtex type: article

% Additional bibtex fields
Bibtex fields:
volume = {5}
number = {2}
pages = {121--125}

Comment:

% One category per line
Categories:
   math/fractals/papers
   by_author/flandrin
```

For PDF files, some of these fields might be automatically filled by `td-add-doc`.

2.3 Configuration file

Start by using the sample configuration file:

```
# Destination path for doc uploads. May be a directory on a remote host.
upload_dest_path = "/tmp/tidydoc"

# File containing the categories list.
# td-generate creates a categories file into output_path.
# This is used by td-add-doc to list existing categories.
categories_path = output_path + '/td_categories'

# Command which will be executed to upload a document into its first category.
# Some variables will be substituted:
# %(destdir)s : upload_dest_path / category
# %(orig_doc_path)s : path towards the input document
# %(orig_dsc_path)s : path towards the input description
# %(final_doc_name)s : destination name of the document
# %(final_dsc_name)s : destination name of the description
upload_commands = \
```

```
"""
mkdir -p "%(destdir)s" || exit 1
cp "%(orig_doc_path)s" "%(destdir)s"/"%(final_doc_name)s" || exit 1
cp "%(orig_dsc_path)s" "%(destdir)s"/"%(final_dsc_name)s" || exit 1
"""

# Command which will be executed to upload a document into other categories.
# Some variables will be substituted:
# %(destdir)s : upload_dest_path / category
# %(rel_first_destdir)s : path towards the directory of the first category
# %(final_doc_name)s : destination name of the document
# %(final_dsc_name)s : destination name of the description
link_commands = \
"""
mkdir -p "%(destdir)s" || exit 1
ln -sf "%(rel_first_destdir)s/%(final_doc_name)s" "%(destdir)s"/"%(final_doc_name)s" |
ln -sf "%(rel_first_destdir)s/%(final_dsc_name)s" "%(destdir)s"/"%(final_dsc_name)s" |
"""

# Commands which will be executed when a document is to be added.
# The document will then be uploaded.
# The following variables are substituted:
# %(orig_doc_path)s : path of the input document
# %(tmp_dsc_path)s : path of the temporary description file already created
add_doc_commands = \
"""
kfmclient exec "%(orig_doc_path)s" &
emacs "%(tmp_dsc_path)s"
"""
```

Configuration variables are Python instructions. Some special variables will be substituted by `tidydoc`.

3 Reorganizing documents

3.1 Using `td-reorganize-doc`

```
Usage: td-reorganize-doc [options] dsc_file

  Options

  -c, --config configfile  Config file to use.
  -v, --version            Print version number.
```

`td-reorganize-doc` takes a description file, and modify the corresponding document organization accordingly. A typical use case is to modify a `.dsc` file, e.g. changing the title and adding a new category, and then calling `td-reorganize-doc` to rename the file and link it into the new categories.

3.2 Additional notes

`td-reorganize-doc` must be use in a machine where the `input_path` is available through the file system, i.e. not a machine which uses `ssh` to upload documents.

4 Generating HTML

4.1 How it works

`td-generate` will walk through the documentation tree and create HTML files for each directory. It looks for `.xml` files, which describe documents. If a document wants to be indexed, an associated `.xml` file have to be crated. XML files can also describe directories, not only files.

`td-generate` will also look for `.dsc` files, and convert them into `.xml` files.

4.2 XML descriptions

Here is a sample XML description, corresponding to the previous `.dsc` file.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
  <document>
    <title>Fractals are everywhere in the world</title>
        <file>foobar.89.micai.fractals_are_everywhere_in_the_world.pdf</file>
        <nbpages>10</nbpages>
        <type>public</type>
        <author>Patrick Foobar</author>
        <date>1989</date>
        <language>english</language>
        <summary>Fractional Brownian motions number of physical phenomena&lt;br&gt;
fractional order. However, the precise meaning of such complementary&lt;br&gt;
approaches are The first one, based on nonstationary nature of&lt;br&gt;
measurements; the second one, self-similarity properties of FBM and&lt;br&gt;
reveals an underlying stationary structure relative to each&lt;br&gt;
time-scaling.</summary>
        <comment><content></content></comment>
        <url>http://patrick.foobar.com/fractals.pdf</url>
      </document>
```

4.3 Using `td-generate`

```
Usage: td-generate [options]

  Options

  -c, --config configfile  Config file to use.
  -v, --version            Print version number.
```

`td-generate` will read the configuration file, and then recursively creates HTML files. If you want `td-generate` not to go into a particular directory, create a `.td_raw_dir` file into it.

```
$ cd my_docs/mess
$ touch .td_raw_dir
```

This way `td-generate` will not consider this directory.

4.4 Configuration file

The default configuration file is commented, here is an example:

```
# Path where document, xml and dsc files are located.
input_path     = "/tmp/tidydoc"

# Path where generated files should be put.
output_path    = "/tmp/tidydoc"

# Template files path.
templates_path = "/usr/local/share/tidydoc/templates"

# Root url of the website. Only useful with html.
site_root      = "file:///tmp/tidydoc"

## HTML parameters

# HTML files which have to be generated at the site root.
site_root_html_files = ["nav.html"]

# HTML files which have to be generated for each directory.
directories_html_files = ["index.html", "doclist.html"]

# Treeview file to link for each directory.
treeview_subdirs_link = "doclist.html"

# Whether treeview should be in multiframe mode or not.
treeview_multiframes = True
```

By using the default HTML template files, this will create documents with a navigation panel on the left frame, and the document descriptions on the right frame.

4.5 Template files

HTML output is driven by template files, which are processed by `td-generate`, by substituting some special variables into it. For example, the template file for `index.html` should be named `index.html.tpl`, and might look like this for a multiframe document:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN">

<html>
<head>
  <title>Documentation</title>
</head>

<frameset cols="35%,65%">
  <frame src="%%SITE_ROOT%%/nav.html" name="treeframe" id="treeframe">
  <frame src="doclist.html" name="basefrm" id="basefrm">
</frameset>
```

```
</html>
```

`%%SITE_ROOT%%` will be substituted by `td-generate`.

Some template files have a special meaning:

- `document.tpl` Processed to create a description for each document.
- `link.tpl` Processed to create a link for each document.
- `subdir.tpl` Processed to create a link for each subdirectory.

4.6 Using Treeview

In order to get a nice tree-like view, Treeview (`http://www.treeview.net`) is supported. To use it, you need to copy the whole directory `doc/examples/treeview` in your `output_path`. This way, `td-generate` will automatically create a `docNodes.js` linking your documents. You can also download directly the archive from the treeview web site.

5 Generating Bibtex

`td-generate` will generate a global bibtex file for all your documents. The output file can be specified in `tidydoc.conf`:

```
## Bibtex parameters

# Output .bib file.
bibtex_file = output_path + "/documents.bib"
```