FreeBSD Documentation Project
Primer for New Contributors
Abstract

Thank you for becoming a part of the FreeBSD Documentation Project. Your contribution is extremely valuable, and we appreciate it.

This primer covers details needed to start contributing to the FreeBSD Documentation Project, or FDP, including tools, software, and the philosophy behind the Documentation Project.

This is a work in progress. Corrections and additions are always welcome.
# Table of Contents

Preface ........................................................................................................... 5

Shell Prompts ................................................................................................ 5

Typographic Conventions ............................................................................ 5

Notes, Tips, Important Information, Warnings, and Examples .................. 5

Acknowledgments ......................................................................................... 6

1. Overview ..................................................................................................... 7

1.1. Documentation in the FreeBSD Ecosystem ..................................... 7

1.2. Quick Start .............................................................................................. 8

1.3. The FreeBSD Documentation Set ..................................................... 14

2. Tools ............................................................................................................ 15

2.1. Required Tools ..................................................................................... 15

2.2. Optional Tools ....................................................................................... 15

3. The Working Copy ...................................................................................... 16

3.1. Documentation and Manual Pages .................................................... 16

3.2. Choosing a Directory ............................................................................ 16

3.3. Checking Out a Copy ............................................................................ 16

3.4. Updating a Working Copy .................................................................... 17

3.5. Reverting Changes .............................................................................. 17

3.6. Making a Diff ......................................................................................... 17

3.7. Git References ....................................................................................... 18

4. Documentation Directory Structure ...................................................... 19

4.1. The Top Level, doc/ ............................................................................. 19

4.2. The Directories ..................................................................................... 19

4.3. Document-Specific Information ...................................................... 20

4.4. The Books: books/ ............................................................................. 20

4.5. The Articles: articles/ ......................................................................... 21

5. The FreeBSD Documentation Build Process ....................................... 23

5.1. Rendering AsciiDoc into Output ....................................................... 23

5.2. The FreeBSD Documentation Build Toolset ................................ 25

5.3. Understanding the Makefile in the Documentation Tree ............. 25

6. AsciiDoctor Primer .................................................................................... 35

6.1. Overview ............................................................................................... 35

6.2. Headings ............................................................................................... 35

6.3. Paragraphs ............................................................................................. 36

6.4. Lists ....................................................................................................... 36

6.5. Links ....................................................................................................... 37

6.6. Conclusion ............................................................................................. 38

7. Rosetta Stone .............................................................................................. 39
Preface

Shell Prompts

This table shows the default system prompt and superuser prompt. The examples use these prompts to indicate which type of user is running the example.

<table>
<thead>
<tr>
<th>User</th>
<th>Prompt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal user</td>
<td>%</td>
</tr>
<tr>
<td>root</td>
<td>#</td>
</tr>
</tbody>
</table>

Typographic Conventions

This table describes the typographic conventions used in this book.

<table>
<thead>
<tr>
<th>Meaning</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>The names of commands.</td>
<td>Use <code>ls -l</code> to list all files.</td>
</tr>
<tr>
<td>The names of files.</td>
<td>Edit <code>.login</code>.</td>
</tr>
<tr>
<td>On-screen computer output.</td>
<td>You have mail.</td>
</tr>
</tbody>
</table>
| What the user types, contrasted with on-screen computer output. | `% date +"The time is %H:%M"
The time is 09:18` |
| Manual page references.                             | Use `su(1)` to change user identity.          |
| User and group names.                               | Only `root` can do this.                      |
| Emphasis.                                           | The user *must* do this.                     |
| Text that the user is expected to replace with the actual text. | To search for a keyword in the manual pages, type `man -k keyword` |
| Environment variables.                              | `$HOME` is set to the user’s home directory.  |

Notes, Tips, Important Information, Warnings, and Examples

Notes, warnings, and examples appear within the text.

Notes are represented like this, and contain information to take note of, as it may affect what the user does.
Tips are represented like this, and contain information helpful to the user, such as showing an easier way to do something.

Important information is represented like this. Typically, these show extra steps the user may need to take.

Warnings are represented like this, and contain information warning about possible damage if the instructions are not followed. This damage may be physical, to the hardware or the user, or it may be non-physical, such as the inadvertent deletion of important files.

Example 1. A Sample Example

Examples are represented like this, and typically contain examples showing a walkthrough, or the results of a particular action.

Acknowledgments

My thanks to Sue Blake, Patrick Durusau, Jon Hamilton, Peter Flynn, and Christopher Maden, who took the time to read early drafts of this document and offer many valuable comments and criticisms.
Chapter 1. Overview

Welcome to the FreeBSD Documentation Project (FDP). Quality documentation is crucial to the success of FreeBSD, and we value your contributions very highly.

This document describes how the FDP is organized, how to write and submit documentation, and how to effectively use the available tools.

Everyone is welcome to contribute to the FDP. Willingness to contribute is the only membership requirement.

This primer shows how to:

• Understand the role of documentation and its place in the ecosystem.
• Identify which parts of FreeBSD are maintained by the FDP.
• Install the required documentation tools and files.
• Make changes to the documentation.
• Submit changes back for review and inclusion in the FreeBSD documentation.

1.1. Documentation in the FreeBSD Ecosystem

All documents are for the benefit of their readers, not their writers or caretakers. They should adapt to the reader and not expect the reader to adapt to them.

Never blame the reader for:

• being unable to make use of a document easily or at all
• finding a document confusing
• not understanding a document or how to apply it
• not finding an explicit answer or successfully bridging gaps (or connecting dots) to reason their way to one

Instead, acknowledge that the document is:

• inaccessible
• confusing
• hard to understand or apply
• incomplete

Then, make the document:

• more accessible
• less confusing
• clearer
• more complete

Use the following methods:

• apply accessibility best practices to correct the problem reported and any similar ones you find
• rework or clarify the confusing structure or language
• add relevant examples to the part that is hard to understand or apply
• fill in the gaps or add the missing stepping stones

1.2. Quick Start

Some preparatory steps must be taken before editing the FreeBSD documentation. First, subscribe to the FreeBSD documentation project mailing list. Some team members also interact on the #bsddocs IRC channel on EFnet. These people can help with questions or problems involving the documentation.

1.2.1. FreeBSD installation process
1. Install these packages. The `docproj meta-port` installs all the applications required to do useful work with the FreeBSD documentation.

```
# pkg install docproj
```

2. Install a local working copy of the documentation from the FreeBSD repository in `~/doc` (see The Working Copy).

```
% git clone https://git.FreeBSD.org/doc.git ~/doc
```

3. Edit the documentation files that require changes. If a file needs major changes, consult the mailing list for input.

Review the output and edit the file to fix any problems shown, then rerun the command to find any remaining problems. Repeat until all of the errors are resolved.

4. **Always** build and test the changes before submitting them. Running `make` in the top-level directory of the documentation will generate that documentation in HTML format.

```
% make
```

5. Add all the files with `git add .`, then review the diff with `git diff`. For example:

```
% git add .
% git diff --staged
```

Make sure that all required files are included, then commit the change to your local branch and generate a patch with `git format-patch`

```
% git commit
% git format-patch origin/main
```

Patch generated with `git format-patch` will include author identity and email addresses, making it easier for developers to apply (with `git am`) and give proper credit.

⚠️ To make it easier for committers to apply the patch on their working copy of the documentation tree, please generate the `.diff` from the base of your documentation tree.

In the example above, changes have been made to the `bsdinstall` portion of the Handbook.

1.2.2. GNU/Linux installation process
Hugo version 0.90 or higher must be used

1. Install these packages in apt-based systems like Debian or Ubuntu. On other GNU/Linux
distributions the package names may change. Consult your distribution's package manager
if in doubt.

```
# apt install hugo ruby-asciidoc ruby-asciidoc-pdf ruby-rouge git bmake
```

2. Install a local working copy of the documentation from the FreeBSD repository in `~/doc`
(see The Working Copy).

```
% git clone https://git.FreeBSD.org/doc.git ~/doc
```

3. Edit the documentation files that require changes. If a file needs major changes, consult the
mailing list for input.

Review the output and edit the files to fix any problems shown, then rerun the command to
find any remaining problems. Repeat until all of the errors are resolved.

4. Always build and test the changes before submitting them. Running `bmake` in the top-level
directory of the documentation will generate that documentation in HTML format.

```
% bmake run LOCALBASE=/usr
```

5. Add all the files with `git add .`, then review the diff with `git diff`. For example:

```
% git add .
% git diff --staged
```

Make sure that all required files are included, then commit the change to your local branch
and generate a patch with `git format-patch`

```
% git commit
% git format-patch origin/main
```

Patch generated with `git format-patch` will include author identity and email addresses,
making it easier for developers to apply (with `git am`) and give proper credit.

To make it easier for committers to apply the patch on their working copy
of the documentation tree, please generate the .diff from the base of your
documentation tree.

6. Submit the diff file using the web-based Problem Report system. If using the web form,
enter a Summary of short description of problem. Select the Component Documentation. In
the Description field, enter a short description of the problem in the Summary field and
add patch to the Keywords field. Use the [Add an attachment] button to attach the diff
file. Finally, use the [Submit Bug] button to submit your diff to the problem report system.

1.2.3. macOS® installation process
1. Install these packages using Homebrew and RubyGem.

   ```
   $ brew install hugo ruby git bmake asciidoctor
   ```

2. Add Ruby to the Path.

   ```
   $ echo 'export PATH="/usr/local/opt/ruby/bin:$PATH"' >> ~/.zshrc
   ```

3. Install the rouge package using RubyGem.

   ```
   $ sudo gem install rouge
   ```

4. Install a local working copy of the documentation from the FreeBSD repository in ~/doc (see The Working Copy).

   ```
   $ git clone https://git.FreeBSD.org/doc.git ~/doc
   ```

5. Edit the documentation files that require changes. If a file needs major changes, consult the mailing list for input.

   Review the output and edit the files to fix any problems shown, then rerun the command to find any remaining problems. Repeat until all of the errors are resolved.

6. Always build and test the changes before submitting them. Running `bmake` in the top-level directory of the documentation will generate that documentation in HTML format.

   ```
   $ bmake run LOCALBASE=/usr
   ```

7. Add all the files with `git add .`, then review the diff with `git diff`. For example:

   ```
   % git add .
   % git diff --staged
   ```

   Make sure that all required files are included, then commit the change to your local branch and generate a patch with `git format-patch`

   ```
   % git commit
   % git format-patch origin/main
Patch generated with `git format-patch` will include author identity and email addresses, making it easier for developers to apply (with `git am`) and give proper credit.

To make it easier for committers to apply the patch on their working copy of the documentation tree, please generate the `.diff` from the base of your documentation tree.

8. Submit the diff file using the web-based Problem Report system. If using the web form, enter a Summary of short description of problem. Select the Component Documentation. In the Description field, enter a short description of the problem in the Summary field and add patch to the Keywords field. Use the [Add an attachment] button to attach the diff file. Finally, use the [Submit Bug] button to submit your diff to the problem report system.

1.3. The FreeBSD Documentation Set

The FDP is responsible for four categories of FreeBSD documentation.

- **Handbook**: The Handbook is the comprehensive online resource and reference for FreeBSD users.
- **FAQ**: The FAQ uses a short question and answer format to address questions that are frequently asked on the various mailing lists and forums devoted to FreeBSD. This format does not permit long and comprehensive answers.
- **Manual pages**: The English language system manual pages are usually not written by the FDP, as they are part of the base system. However, the FDP can reword parts of existing manual pages to make them clearer or to correct inaccuracies.
- **Web site**: This is the main FreeBSD presence on the web, visible at https://www.FreeBSD.org/ and many mirrors around the world. The web site is typically a new user’s first exposure to FreeBSD.

Translation teams are responsible for translating the Handbook and web site into different languages. Manual pages are not translated at present.

Documentation source for the FreeBSD web site, Handbook, and FAQ is available in the documentation repository at https://cgit.freebsd.org/doc/.

Source for manual pages is available in a separate source repository located at https://cgit.freebsd.org/src/.

Documentation commit messages are visible with `git log`. Commit messages are also archived at Commit messages for all branches of the doc repository.

Web frontends to both of these repositories are available at https://cgit.freebsd.org/doc/ and https://cgit.freebsd.org/src/.

Many people have written tutorials or how-to articles about FreeBSD. Some are stored as part of the FDP files. In other cases, the author has decided to keep the documentation separate. The FDP endeavors to provide links to as much of this external documentation as possible.
Chapter 2. Tools

Several software tools are used to manage the FreeBSD documentation and render it to different output formats. Some of these tools are required and must be installed before working through the examples in the following chapters. Some are optional, adding capabilities or making the job of creating documentation less demanding.

2.1. Required Tools

Install `docproj metaport` as shown in the overview chapter from the Ports Collection. These applications are required to do useful work with the FreeBSD documentation. Some further notes on particular components are given below.

2.2. Optional Tools

These applications are not required, but can make working on the documentation easier or add capabilities.

2.2.1. Software

**Vim** (editors/vim)

A popular editor for working with AsciiDoctor.

**Emacs** (editors/emacs)

Both of these editors include a special mode for editing documents. This mode includes commands to reduce the amount of typing needed, and help reduce the possibility of errors.
Chapter 3. The Working Copy

The *working copy* is a copy of the FreeBSD repository documentation tree downloaded onto the local computer. Changes are made to the local working copy, tested, and then submitted as patches to be committed to the main repository.

A full copy of the documentation tree can occupy 550 megabytes of disk space. Allow for a full gigabyte of space to have room for temporary files and test versions of various output formats.

Git is used to manage the FreeBSD documentation files. It is obtained by installing the Git package:

```
# pkg install git-lite
```

### 3.1. Documentation and Manual Pages

FreeBSD documentation is not just books and articles. Manual pages for all the commands and configuration files are also part of the documentation, and part of the FDP’s territory. Two repositories are involved: *doc* for the books and articles, and *src* for the operating system and manual pages. To edit manual pages, the *src* repository must be checked out separately.

Repositories may contain multiple versions of documentation and source code. New modifications are almost always made only to the latest version, called *main*.

### 3.2. Choosing a Directory

FreeBSD documentation is traditionally stored in `/usr/doc/`, and system source code with manual pages in `/usr/src/`. These directory trees are relocatable, and users may want to put the working copies in other locations to avoid interfering with existing information in the main directories. The examples that follow use `~/doc` and `~/src`, both subdirectories of the user’s home directory.

### 3.3. Checking Out a Copy

A download of a working copy from the repository is called a *clone*, and done with `git clone`. This example clones a copy of the latest version (*main*) of the main documentation tree:

```
% git clone https://git.FreeBSD.org/doc.git ~/doc
```

A checkout of the source code to work on manual pages is very similar:

```
% git clone https://git.FreeBSD.org/src.git ~/src
```
3.4. Updating a Working Copy

The documents and files in the FreeBSD repository change daily. People modify files and commit changes frequently. Even a short time after an initial checkout, there will already be differences between the local working copy and the main FreeBSD repository. To update the local version with the changes that have been made to the main repository, use `git pull` on the directory containing the local working copy:

```
% cd ~/doc
% git pull --ff-only
```

Get in the protective habit of using `git pull` before editing document files. Someone else may have edited that file very recently, and the local working copy will not include the latest changes until it has been updated. Editing the newest version of a file is much easier than trying to combine an older, edited local file with the newer version from the repository.

3.5. Reverting Changes

Sometimes it turns out that changes were not necessary after all, or the writer just wants to start over. Files can be "reset" to their unchanged form with `git restore`. For example, to erase the edits made to `_index.adoc` and reset it to unmodified form:

```
% git restore _index.adoc
```

3.6. Making a Diff

After edits to a file or group of files are completed, the differences between the local working copy and the version on the FreeBSD repository must be collected into a single file for submission. These `diff` files are produced by redirecting the output of `git diff` into a file:

```
% cd ~/doc
% git diff > doc-fix-spelling.diff
```

Give the file a meaningful name that identifies the contents. The example above is for spelling fixes to the whole documentation tree.

If the diff file is to be submitted with the web "Submit a FreeBSD problem report" interface, add a `.txt` extension to give the earnest and simple-minded web form a clue that the contents are plain text.

Be careful: `git diff` includes all changes made in the current directory and any subdirectories. If there are files in the working copy with edits that are not ready to be submitted yet, provide a list of only the files that are to be included:
3.7. Git References

These examples show very basic usage of Git. More detail is available in the Git Book and the Git documentation.
Chapter 4. Documentation Directory Structure

Files and directories in the doc/ tree follow a structure meant to:

1. Make it easy to automate converting the document to other formats.
2. Promote consistency between the different documentation organizations, to make it easier to switch between working on different documents.
3. Make it easy to decide where in the tree new documentation should be placed.

In addition, the documentation tree must accommodate documents in many different languages. It is important that the documentation tree structure does not enforce any particular defaults or cultural preferences.

4.1. The Top Level, doc/

There are three sections under doc/, documentation and website share the same structure.

<table>
<thead>
<tr>
<th>Directory</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>documentation</td>
<td>Contains all the articles and books in AsciiDoc format. Contains subdirectories to further categorize the information by languages.</td>
</tr>
<tr>
<td>tools</td>
<td>Contains a set of tools used to translate the documentation and the website using Weblate. The Weblate instance can be found here.</td>
</tr>
<tr>
<td>shared</td>
<td>Contains files that are not specific to the various translations of the documentation. Contains subdirectories to further categorize the information by languages and three files to store the authors, releases and mirrors information. This directory is shared between documentation and the website.</td>
</tr>
<tr>
<td>website</td>
<td>Contains the FreeBSD website in AsciiDoc format. Contains subdirectories to further categorize the information by languages.</td>
</tr>
</tbody>
</table>

4.2. The Directories

These directories contain the documentation and the website. The documentation is organized into subdirectories below this level, following the Hugo directory structure.

<table>
<thead>
<tr>
<th>Directory</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>archetypes</td>
<td>Contain templates to create new articles, books and webpages. For more information take a look here.</td>
</tr>
<tr>
<td>config</td>
<td>Contain the Hugo configuration files. One main file and one file per language. For more information take a look here.</td>
</tr>
<tr>
<td>content</td>
<td>Contain the books, articles and webpages. One directory exists for each available translation of the documentation, for example en and zh-tw.</td>
</tr>
<tr>
<td>Directory</td>
<td>Usage</td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td>data</td>
<td>Contain custom data for build the website in TOML format. This directory is used to store the events, news, press, etc. For more information take a look here.</td>
</tr>
<tr>
<td>static</td>
<td>Contain static assets. Images, security advisories, the pgpkeys, etc. For more information take a look here.</td>
</tr>
<tr>
<td>themes</td>
<td>Contain the templates in the form of .html files that specify how the website looks. For more information take a look here.</td>
</tr>
<tr>
<td>tools</td>
<td>Contain tools used to enhance the documentation build. For example to generate the Table of Contents of the books, etc.</td>
</tr>
<tr>
<td>beastie.png</td>
<td>This image doesn’t need an introduction ;)</td>
</tr>
<tr>
<td>LICENSE</td>
<td>License of the documentation, shared and website. BSD 2-Clause License.</td>
</tr>
<tr>
<td>Makefile</td>
<td>The Makefile defines the build process of the documentation and the website.</td>
</tr>
</tbody>
</table>

4.3. Document-Specific Information

This section contains specific notes about particular documents managed by the FDP.

4.4. The Books: books/

The books are written in AsciiDoc.

The books are organized as an AsciiDoc book. The books are divided into parts, each of which contains several chapters. Chapters are further subdivided into sections (=) and subsections (==, ===) and so on.

4.4.1. Physical Organization

There are a number of files and directories within the books directory, all with the same structure.

4.4.1.1. _index.adoc

The _index.adoc defines some AsciiDoc variables that affect how the AsciiDoc source is converted to other formats and list the Table of Contents, Table of Examples, Table of Figures, Table of Tables and the abstract section.

4.4.1.2. book.adoc

The _index.adoc defines some AsciiDoc variables that affect how the AsciiDoc source is converted to other formats and list the Table of Contents, Table of Examples, Table of Figures, Table of Tables, the abstract section and all the chapters. This file is used to generate the PDF with asciidoctor-pdf and to generate the book in one html page.
4.4.1.3. part*.adoc

The part.adoc* files stores a brief introduction of one part of the book.

4.4.1.4. directory/_index.adoc

Each chapter in the Handbook is stored in a file called _index.adoc in a separate directory from the other chapters.

For example, this is an example of the header of one chapter:

```adoc
---
title: Chapter 8. Configuring the FreeBSD Kernel
part: Part II. Common Tasks
prev: books/handbook/multimedia
next: books/handbook/printing
---

[[kernelconfig]]
= Configuring the FreeBSD Kernel
...
```

When the HTML5 version of the Handbook is produced, this will yield kernelconfig/index.html.

A brief look will show that there are many directories with individual _index.adoc files, including basics/_index.adoc, introduction/_index.adoc, and printing/_index.adoc.

⚠️ Do not name chapters or directories after their ordering within the Handbook. This ordering can change as the content within the Handbook is reorganized. Reorganization should be possible without renaming files, unless entire chapters are being promoted or demoted within the hierarchy.

4.5. The Articles: articles/

The articles are written in AsciiDoc.

The articles are organized as an AsciiDoc article. The articles are divided into sections (=) and subsections (==, ===) and so on.

4.5.1. Physical Organization

There is one _index.adoc file per article.

4.5.1.1. _index.adoc

The _index.adoc file contains all the AsciiDoc variables and the content.

For example, this is an example of one article, the structure is pretty similar to one book chapter:
Why you should use a BSD style license for your Open Source Project

Introduction
Chapter 5. The FreeBSD Documentation Build Process

This chapter covers organization of the documentation build process and how `make(1)` is used to control it.

5.1. Rendering AsciiDoc into Output

Different types of output can be produced from a single AsciiDoc source file.

<table>
<thead>
<tr>
<th>Formats</th>
<th>File Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>html</td>
<td>HTML</td>
<td>An article or book chapter.</td>
</tr>
<tr>
<td>pdf</td>
<td>PDF</td>
<td>Portable Document Format.</td>
</tr>
<tr>
<td>epub</td>
<td>EPUB</td>
<td>Electronic Publication. ePub file format.</td>
</tr>
</tbody>
</table>

5.1.1. Rendering to html

To render the documentation and the website to html use one of the following examples.

*Example 2. Build the documentation*

```
% cd ~/doc/documentation
% make
```

*Example 3. Build the website*

```
% cd ~/doc/website
% make
```

*Example 4. Build the entire documentation project*

```
% cd ~/doc
% make -j2
```

Advanced build examples are given below:
Example 5. Build the documentation with verbose and debug messages

```bash
% cd ~/doc/documentation
% make HUGO_ARGS="--verbose --debug --path-warnings"
```

Example 6. Build and serve the content with Hugo's internal webserver

```bash
% cd ~/doc/documentation
% make run
```

This webserver runs on `localhost`, port **1313** by default.

To serve the content with Hugo’s internal webserver binding a specific IP address:

```bash
% make run BIND=192.168.15.10
```

A `hostname` can also be set as base url to Hugo’s internal webserver:

```bash
% make run BIND=192.168.15.10 HOSTNAME=example.com
```

### 5.1.2. Rendering to pdf

To render the documentation to **pdf**, use one of the following examples.

**Example 7. Build all documents in pdf**

```bash
% cd ~/doc/documentation
% make pdf
```

**Example 8. Build all articles in pdf**

```bash
% cd ~/doc/documentation
% make pdf-articles
```
Example 9. Build all books in pdf

```bash
% cd ~/doc/documentation
% make pdf-books
```

Example 10. Build documents in pdf for specific languages

```bash
% cd ~/doc/documentation
% make DOC_LANG="en" pdf
```

This will build all English documents in pdf.

```bash
% cd ~/doc/documentation
% make DOC_LANG="en fr" pdf-books
```

This will build all English and French books in pdf.

5.2. The FreeBSD Documentation Build Toolset

These are the tools used to build and install the FDP documentation.

- The primary build tool is `make(1)`, specifically Berkeley Make.
- Hugo
- AsciiDoctor
- Git

5.3. Understanding the Makefile in the Documentation Tree

There are three Makefile files for building some or all of the documentation project.

- The Makefile in the documentation directory will build only the documentation.
- The Makefile in the website directory will build only the website.
- The Makefile at the top of the tree will build both the documentation and the website.

The Makefile appearing in subdirectories also support `make run` to serve built content with Hugo’s internal webserver. This webserver runs on port 1313 by default.

5.3.1. Documentation Makefile

This Makefile takes the following form:
# Generate the FreeBSD documentation
#
# Copyright (c) 2020-2021, The FreeBSD Documentation Project
# Copyright (c) 2020-2021, Sergio Carlavilla <carlavilla@FreeBSD.org>
#
# Targets intended for use on the command line
#
# all (default) -   generate the books TOC and compile all the documentation
# clean     -   removes generated files
# run       -   serves the built documentation site for local browsing
# pdf       -   build PDF versions of the articles and books.
# html      -   build HTML versions of the articles and books for offline use.
#            If variable DOC_HTML_ARCHIVE is set, all documents will be archived/compressed, and only these files will be kept in the public directory.
# epub      -   build EPUB versions of the articles and books (Experimental).
#
# The run target uses hugo's built-in webserver to make the documentation site available for local browsing. The documentation should have been built prior to attempting to use the `run` target. By default, hugo will start its webserver on port 1313.

MAINTAINER=carlavilla@FreeBSD.org ①

# List of languages without book translations
ARTICLEONLY_LANGS= bn-bd da ko tr

# List of languages without article translations
BOOKONLY_LANGS= mn

# List of all languages we have content for
ALL_LANGUAGES= bn-bd da de el en es fr hu it ja ko mn nl pl pt-br ru tr zh-cn zh-tw ②

LOCALBASE?= /usr/local

RUBY_CMD = ${LOCALBASE}/bin/ruby ③
HUGO_CMD = ${LOCALBASE}/bin/hugo ④
HUGO_ARGS?= --verbose --minify
HUGO_OFFLINE_ARGS?= --environment offline --verbose --minify
ASCIIODOCTOR_CMD= ${LOCALBASE}/bin/asciidoctor
ASCIIODOCTORPDF_CMD= ${LOCALBASE}/bin/asciidoctor-pdf

.ifdef Defined(DOC_LANG) && !empty(DOC_LANG)
LANGUAGES= ${DOC_LANG:/:// /g}
.ifdef $(LANGUAGES:Men) == "" && $(TARGETS:Mpdf*) == "" && $(TARGETS:Mhtml*) == ""
.warning "Warning: cannot skip 'en'; adding it back"
LANGUAGES+= en
.endif
.else

26
LANGUAGES= \${ALL_LANGUAGES}
.endif

RUBYLIB = \./shared/lib
.export RUBYLIB

RUN_DEPENDS= \${HUGO_CMD} \n \${LOCALBASE}/bin/asciidoctor \n \${LOCALBASE}/bin/rougify

ifndef HOSTNAME
.ifdef BIND
.HOST=$(BIND)
.else
.HOST=localhost
.endif
else
.HOST=$(HOSTNAME)
.endif

# Strip the languages with only articles from the list of languages we
# will use to build books.
BOOK_LANGS= \${LANGUAGES}
.for a in \${ARTICLEONLY_LANGS}
BOOK_LANGS+= \${BOOK_LANGS}:N\${a}
.endfor

# Strip the languages with only books from the list of languages we
# will use to build articles.
ARTICLE_LANGS= \${LANGUAGES}
.for a in \${BOOKONLY_LANGS}
ARTICLE_LANGS+= \${ARTICLE_LANGS}:N\${a}
.endfor

# Take the list of all languages, and take out the ones we have been
# asked for. We'll feed this to hugo.
SKIP_LANGS= \n.for a in \${ALL_LANGUAGES}
.if \${LANGUAGES}:M\${a} == ""
SKIP_LANGS+= \${a}
.endif
.endfor

.ORDER: all run ⑤

.ORDER: requirements ⑥
.ORDER: starting-message
.ORDER: starting-message build
.ORDER: build

all: requirements starting-message generate-pgpkeys-txt build
run: requirements starting-message generate-pgpkeys-txt run-local

# clean does not call pdf-clean as that is a subset of hugo-clean

clean: hugo-clean ppg-clean

requirements:
  .for dep in ${RUN_DEPENDS}
  .if !exists(${dep})
    @(echo ${dep} not found, please run 'pkg install docproj'; exit 1)
  .endif
  .endfor

requirements-pdf:
  .if !exists(${LOCALBASE}/bin/asciidoc-pdf)
    @(echo ${LOCALBASE}/bin/asciidoc-pdf not found, please run 'pkg install rubygem-asciidoc-pdf'; exit 1)
  .endif

requirements-epub:
  .if !exists(${LOCALBASE}/bin/asciidoc-epub3)
    @(echo ${LOCALBASE}/bin/asciidoc-epub3 not found, please run 'pkg install rubygem-asciidoc-epub3'; exit 1)
  .endif

starting-message: .PHONY ⑦
  @echo ---------------------------------------------------------------
  @echo                   Building the documentation
  @echo  included languages: ${LANGUAGES}
  @echo  excluded languages: ${SKIP_LANGS}
  @echo ---------------------------------------------------------------

generate-pgpkeys-txt: static/pgpkeys/pgpkeys.txt

static/pgpkeys/pgpkeys.txt: static/pgpkeys/*key
  ${RUBY_CMD} ./tools/global-pgpkeys-creator.rb

run-local: .PHONY ⑧
  HUGO_DISABLELANGUAGES=${SKIP_LANGS} ${HUGO_CMD} server \$
  ${HUGO_ARGS} --D $(BIND:D--bind=$(BIND)) --baseURL="http://$(.HOST):1313"

build: .PHONY ⑨
  HUGO_DISABLELANGUAGES=${SKIP_LANGS} ${HUGO_CMD} ${HUGO_ARGS}

build-offline: .PHONY
  HUGO_DISABLELANGUAGES=${SKIP_LANGS} ${HUGO_CMD} ${HUGO_OFFLINE_ARGS}

ppg-clean: .PHONY
  rm -f static/pgpkeys/pgpkeys.txt

hugo-clean: .PHONY
  rm -rf resources public
# PDF targets
# Use DOC_LANG to choose the language, e.g., make DOC_LANG="en fr" pdf-books
#
pdf: pdf-articles pdf-books

pdf-books: requirements-pdf
  .for _lang in ${BOOK_LANGS}
    ./tools/asciidoctor.sh books ${_lang} pdf
  .endfor

pdf-articles: requirements-pdf
  .for _lang in ${ARTICLE_LANGS}
    ./tools/asciidoctor.sh articles ${_lang} pdf
  .endfor

pdf-clean: pdf-articles-clean pdf-books-clean

pdf-books-clean:
  .for _lang in ${BOOK_LANGS}
    rm -fr ${CURDIR}/public/${_lang}/books
    -rmdir ${CURDIR}/public/${_lang}
  .endfor
    -rmdir ${CURDIR}/public/

pdf-articles-clean:
  .for _lang in ${ARTICLE_LANGS}
    rm -fr ${CURDIR}/public/${_lang}/articles
    if !exists(${CURDIR}/public/${_lang}/books)
      rm -fr ${CURDIR}/public/${_lang}
    .endif
  .endfor
    -rmdir ${CURDIR}/public/

# HTML targets
#

html-clean: hugo-clean

html-clean-global:
  rm -fr ${CURDIR}/public/index.html
  rm -rf pgpkeys js

html-clean-articles:
  .for _lang in ${ARTICLE_LANGS}
    rm -fr ${CURDIR}/public/${_lang}/index.html
    rm -fr ${CURDIR}/public/${_lang}/articles/index.html
html-clean-books:
.for _lang in ${BOOK_LANGS}
   rm -fr ${.CURDIR}/public/${_lang}/books/index.html
.endfor

html-archive:
.if defined(DOC_HTML_ARCHIVE)
.for _lang in ${ARTICLE_LANGS}
   ./tools/asciidoctor.sh articles ${_lang} archive
.endfor
.for _lang in ${BOOK_LANGS}
   ./tools/asciidoctor.sh books ${_lang} archive
.endfor
.endif

html-archive-clean-files:
.if defined(DOC_HTML_ARCHIVE)
   find ${.CURDIR}/public/ ! -name '*.pdf' ! -name '*.tar.gz' -type f -delete
   find ${.CURDIR}/public/ -type d -empty -delete
.endif

# # EPUB targets
# Use DOC_LANG to choose the language, e.g., make DOC_LANG="en fr" epub-books
#
epub: epub-articles epub-books

epub-books: requirements-epub
   @echo ---------------------------------------------------------------
   @echo !!! EPUB output is experimental !!!
   @echo
   @echo Asciidoctor EPUB3 is currently alpha software. Use accordingly. Although the
   @echo bulk of AsciiDoc content is converted, there's still work needed to fill in
   @echo gaps where conversion is incomplete or unstyled.
   @echo https://docs.asciidoctor.org/epub3-converter/latest/#project-status
   @echo ---------------------------------------------------------------
.for _lang in ${BOOK_LANGS}
   ./tools/asciidoctor.sh books ${_lang} epub
.endfor

epub-articles: requirements-epub
   @echo ---------------------------------------------------------------
   @echo !!! EPUB output is experimental !!!
   @echo
   @echo Asciidoctor EPUB3 is currently alpha software. Use accordingly. Although the
   @echo bulk of AsciiDoc content is converted, there's still work needed to fill in
   @echo gaps where conversion is incomplete or unstyled.
   @echo https://docs.asciidoctor.org/epub3-converter/latest/#project-status
   @echo ---------------------------------------------------------------
The **MAINTAINER** flag specifies who is the maintainer of this Makefile.

**ALL_LANGUAGES** flag specifies in which languages the table of contents has to be generated.

**RUBY_CMD** flag specifies the location of the Ruby binary.

**HUGO_CMD** flag specifies the location of the Hugo binary.

**.ORDER** directives are used to ensure multiple make jobs may run without problem.

**all** target builds the documentation and puts the result in `~/doc/documentation/public`.

**run-local** runs hugo webserver on port 1313, or a random free port if that is already in use.

**build** builds the documentation and puts the result in the `~/doc/documentation/public`.

### 5.3.2. Website Makefile

This Makefile takes the form of:

```make
# Generate the FreeBSD website
#
# Copyright (c) 2020-2021, The FreeBSD Documentation Project
# Copyright (c) 2020-2021, Sergio Carlavilla <carlavilla@FreeBSD.org>
#
# Targets intended for use on the command line
#
# all (default) - generate the releases.toml and compile all the website
# run   - serves the built website for local browsing
#
```
# The run target uses hugo’s built-in webserver to make the built website
# available for local browsing. The website should have been built prior
# to attempting to use the ‘run’ target. By default, hugo will start its
# webserver on port 1313.

MAINTAINER=carlavilla@FreeBSD.org ①

# List of all languages we have content for
ALL_LANGUAGES=  de el en es fr hu it ja nl ru tr zh-cn zh-tw

LOCALBASE?= /usr/local

RUBY_CMD = ${LOCALBASE}/bin/ruby ②
HUGO_CMD = ${LOCALBASE}/bin/hugo ③
HUGO_ARGS?= --verbose
RUBYLIB = ../shared/lib
.export RUBYLIB

ifndef HOSTNAME
  ifdef BIND
    HOST=$(BIND)
  .else
    HOST=localhost
  .endif
.else
  HOST=$(HOSTNAME)
endif

if defined(DOC_LANG) && !empty(DOC_LANG)
LANGUAGES=  ${DOC_LANG}:S/,/g
.if  ${LANGUAGES:M} == ""
.warning "Warning: cannot skip 'en'; adding it back"
LANGUAGES+= en
endif
.else
LANGUAGES=  ${ALL_LANGUAGES}
endif

# Take the list of all languages, and take out the ones we have been
# asked for via DOC_LANG. We'll feed this to hugo.
SKIP_LANGS=
.for a in ${ALL_LANGUAGES}
.if ${LANGUAGES:M${a}} == ""
SKIP_LANGS+=  ${a}
.endif
.endfor

.ORDD: all run ④

.ORDD: starting-message generate-releases
.ORDD: starting-message build
.ORDER: generate-releases build
.ORDER: build post-build
.ORDER: post-build end-message

all: starting-message generate-releases build post-build end-message ⑤
run: starting-message generate-releases run-local
clean: hugo-clean releases-clean

starting-message: .PHONY ⑥
@echo "---------------------------------------------------------------"
@echo "Building the website started on $$\$(date)"
@echo " included languages: $$\${LANGUAGES}"
@echo " excluded languages: $$\${SKIP_LANGS}"
@echo "---------------------------------------------------------------"

data/releases.toml: ⑦

run-local: .PHONY ⑧
  HUGO_DISABLELANGUAGES="\${SKIP_LANGS}" \${HUGO_CMD} server \n  \${HUGO_ARGS} -D \$(BIND:D--bind=$\{(BIND)\}) --baseURL="http://\$(.HOST):1313"

build: .PHONY ⑨
  HUGO_DISABLELANGUAGES="\${SKIP_LANGS}" \${HUGO_CMD} \${HUGO_ARGS}

post-build: cgi-permissions

cgi-permissions:
  @chmod 555 ./public/cgi/*.cgi

hugo-clean:
  rm -fr public resources

releases-clean:
  rm -f data/releases.toml

---

① The MAINTAINER flag specifies who is the maintainer of this Makefile.
② RUBY_CMD flag specifies the location of the Ruby binary.
③ HUGO_CMD flag specifies the location of the Hugo binary.
④ .ORDER directives are used to ensure multiple make jobs may run without problem.
⑤ all target builds the website and puts the result in ~/doc/website/public.
⑥ **starting-message** shows a message in the CLI to show the user that the process is running.

⑦ **generate-releases** calls the script used to convert from AsciiDoc variables to TOML variables. With this conversion, the releases variables can be used in AsciiDoc and in the Hugo custom templates.

⑧ **run-local** runs hugo webserver on port 1313, or a random free port if that is already in use.

⑨ **build** builds the website and puts the result in the `~/doc/website/public`.
Chapter 6. Asciidoctor Primer

Most FDP documentation is written with Asciidoc. This chapter explains what that means, how to read and understand the documentation source, and the techniques used. To get a complete reference of the Asciidoctor capabilities please consult the Asciidoctor documentation. Some of the examples used in this chapter have been taken from the AsciDoc Syntax Quick Reference.

6.1. Overview

In the original days of computers, electronic text was simple. There were a few character sets like ASCII or EBCDIC, but that was about it. Text was text, and what you saw really was what you got. No frills, no formatting, no intelligence.

Inevitably, this was not enough. When text is in a machine-readable format, machines are expected to be able to use and manipulate it intelligently. Authors want to indicate that certain phrases should be emphasized, or added to a glossary, or made into hyperlinks. Filenames could be shown in a “typewriter” style font for viewing on screen, but as “italics” when printed, or any of a myriad of other options for presentation.

It was once hoped that Artificial Intelligence (AI) would make this easy. The computer would read the document and automatically identify key phrases, filenames, text that the reader should type in, examples, and more. Unfortunately, real life has not happened quite like that, and computers still require assistance before they can meaningfully process text.

More precisely, they need help identifying what is what. Consider this text:

To remove /tmp/foo, use `rm(1)`.

```shell
% rm /tmp/foo
```

It is easy for the reader to see which parts are filenames, which are commands to be typed in, which parts are references to manual pages, and so on. But the computer processing the document cannot reliably determine this. For this we need markup.

The previous example is actually represented in this document like this:

```asciidoc

[source,shell]
----
% rm /tmp/foo
----
```

6.2. Headings

Asciidoctor supports six headings levels. If the document type is `article` only one level 0 (=) can be
used. If the document type is `book` then there can be multiple level 0 (=) headings.

This is an example of headings in an article.

```
= Document Title (Level 0)
== Level 1 Section Title
=== Level 2 Section Title
==== Level 3 Section Title
===== Level 4 Section Title
====== Level 5 Section Title
== Another Level 1 Section Title
```

Section levels cannot be skipped when nesting sections.

The following syntax is not correct.
```
= Document Title
== Level 2
==== Level 4
```

6.3. Paragraphs

Paragraphs don't require special markup in AsciiDoc. A paragraph is defined by one or more consecutive lines of text. To create a new paragraph leave one blank line.

For example, this is a heading with two paragraphs.
```
= This is the heading

This is the first paragraph.
This is also the first paragraph.

And this is the second paragraph.
```

6.4. Lists

AsciiDoctor supports a few types of lists, the most common are `ordered` and `unordered`. To get more information about lists, see [AsciiDoc Syntax Quick Reference](#).
6.4.1. Ordered lists

To create an ordered list use the . character.

For example, this is an ordered list.

. First item
. Second item
.. Subsecond item
. Third item

And this would be rendered as.

1. First item
2. Second item
   a. Subsecond item
3. Third item

6.4.2. Unordered lists

To create an unordered list use the * character.

For example, this is an unordered list.

* First item
* Second item
** Subsecond item
* Third item

And this would be rendered as.

• First item
• Second item
  ◦ Subsecond item
• Third item

6.5. Links

6.5.1. External links

To point to another website the link macro should be used.

link:https://www.FreeBSD.org[FreeBSD]
As the AsciiDoctor documentation describes, the link macro is not required when the target starts with a URL scheme like https. However, it is a good practice to do this anyway to ensure that AsciiDoctor renders the link correctly, especially in non-latin languages like Japanese.

6.5.2. Internal link

To point to another book or article the AsciiDoctor variables should be used. For example, if we are in the cups article and we want to point to ipsec-must these steps should be used.

1. Include the urls.adoc file from ~/doc/shared folder.

```
include::shared/{lang}/urls.adoc[]
```

2. Then create a link using the AsciiDoctor variable to the ipsec-must article.

```
extref:{ipsec-must}[IPSec-Must article]
```

And this would be rendered as.

IPSec-Must article

6.6. Conclusion

This is the conclusion of this AsciiDoctor primer. For reasons of space and complexity, several things have not been covered in depth (or at all).
## 7.1. Comparison between Docbook and AsciIDoc

This rosetta stone tries to show the differences between Docbook and AsciIDoc.

<table>
<thead>
<tr>
<th>Language Feature</th>
<th>Docbook</th>
<th>AsciIDoc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bold</td>
<td><code>&lt;strong&gt;bold&lt;/strong&gt;</code></td>
<td><code>*bold*</code></td>
</tr>
<tr>
<td>Italic</td>
<td><code>&lt;emphasis&gt;Italic&lt;/emphasis&gt;</code></td>
<td><code>_Italic_</code></td>
</tr>
<tr>
<td>Monospace</td>
<td><code>&lt;literal&gt;Monospace&lt;/literal&gt;</code></td>
<td>``Monospace`</td>
</tr>
<tr>
<td>Paragraph</td>
<td><code>&lt;para&gt;This is a paragraph&lt;/para&gt;</code></td>
<td>This is a paragraph</td>
</tr>
<tr>
<td>Keycap</td>
<td><code>&lt;keycap&gt;F11&lt;/keycap&gt;</code></td>
<td><code>kbd:[F11]</code></td>
</tr>
<tr>
<td>Links</td>
<td><code>&lt;link xlink:href=&quot;https://www.freebsd.org/where/&quot;&gt;Download FreeBSD&lt;/link&gt;</code></td>
<td><code>link:https://www.freebsd.org/where/ [Download FreeBSD]</code></td>
</tr>
<tr>
<td>Sections</td>
<td><code>&lt;sect1 xml:id=&quot;id&quot;&gt;</code></td>
<td><code>[[id]] = Section 1</code></td>
</tr>
<tr>
<td>Unordered list</td>
<td><code>&lt;itemizedlist&gt;</code></td>
<td>* When to build a custom kernel.</td>
</tr>
<tr>
<td></td>
<td><code> &lt;listitem&gt;</code></td>
<td>* How to take a hardware inventory.</td>
</tr>
<tr>
<td></td>
<td><code>  &lt;para&gt;When to build a custom kernel.&lt;/para&gt;</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code> &lt;/listitem&gt;</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code> &lt;listitem&gt;</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>  &lt;para&gt;How to take a hardware inventory.&lt;/para&gt;</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code> &lt;/listitem&gt;</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code> &lt;/itemizedlist&gt;</code></td>
<td></td>
</tr>
<tr>
<td>Language Feature</td>
<td>Docbook</td>
<td>AsciiDoc</td>
</tr>
<tr>
<td>------------------</td>
<td>----------</td>
<td>----------</td>
</tr>
</tbody>
</table>
| **Ordered list** | ```<orderedlist>
   <listitem>
     <para>One</para>
   </listitem>
   <listitem>
     <para>Two</para>
   </listitem>
   <listitem>
     <para>Three</para>
   </listitem>
   <listitem>
     <para>Four</para>
   </listitem>
</orderedlist>``` | . One  
. Two  
. Three  
. Four |
| **Variable list** | ```<variablelist>
   <varlistentry>
     <term>amd64</term>
     <listitem>
       <para>This is the most common desktop...</para>
     </listitem>
   </varlistentry>
</variablelist>``` | amd64::  
This is the most common desktop... |
| **Source code** | ```<screen>
   &prompt.root; <userinput>mkdir -p /var/spool/lpd/lp</userinput>
</screen>``` | [source,shell]  
```----
# mkdir -p /var/spool/lpd/lp
----``` |
| **Literal block** | ```<programlisting>
include GENERIC
ident MYKERNEL

options IPFIREWALL
options DUMMYNET
options IPFIREWALL_DEFAULT_TO_ACCEPT
options IPDIVERT
</programlisting>``` | ....  
include GENERIC  
ident MYKERNEL  
```
options IPFIREWALL
options DUMMYNET
options IPFIREWALL_DEFAULT_TO_ACCEPT
options IPDIVERT
....``` |
<table>
<thead>
<tr>
<th>Language Feature</th>
<th>Docbook</th>
<th>Asciidoc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Images</td>
<td><code>&lt;figure xml:id=&quot;bsdinstall-newboot-loader-menu&quot;&gt; &lt;title&gt;FreeBSD Boot Loader Menu&lt;/title&gt; &lt;mediaobject&gt; &lt;imageobject&gt; &lt;imagedata fileref=&quot;bsdinstall/bsdinstall-newboot-loader-menu&quot;/&gt; &lt;/imageobject&gt; &lt;textobject&gt; &lt;/literallayout&gt;ASCII art replacement is no longer supported.&lt;/literallayout&gt; &lt;/textobject&gt; &lt;textobject&gt; &lt;phrase&gt;The FreeBSD loader menu, with options 1-6 to boot multi-user, boot single user, escape to loader prompt, reboot, select a kernel to load, and select boot options&lt;/phrase&gt; &lt;/textobject&gt; &lt;/mediaobject&gt; &lt;/figure&gt;</code></td>
<td><code>[[bsdinstall-newboot-loader-menu]] FreeBSD Boot Loader Menu image::bsdinstall/bsdinstall-newboot-loader-menu[The FreeBSD loader menu, with options 1-6 to boot multi-user, boot single user, escape to loader prompt, reboot, select a kernel to load, and select boot options]</code></td>
</tr>
<tr>
<td>Includes</td>
<td>n/a</td>
<td>include::chapter.adoc[]</td>
</tr>
</tbody>
</table>
### Tables

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APM</td>
<td>Apple Partition Map, used by PowerPC(R).</td>
</tr>
</tbody>
</table>

**Admonitions**

- **tip**
  - This is a tip

- **TIP**
  - This is a tip
Chapter 8. Translations

This is the FAQ for people translating the FreeBSD documentation (FAQ, Handbook, tutorials, manual pages, and others) to different languages.

It is very heavily based on the translation FAQ from the FreeBSD German Documentation Project, originally written by Frank Gründer elwood@mc5sys.in-berlin.de and translated back to English by Bernd Warken bwarken@mayn.de.

8.1. What do i18n and l10n mean?

i18n means internationalization and l10n means localization. They are just a convenient shorthand.

i18n can be read as "i" followed by 18 letters, followed by "n". Similarly, l10n is "l" followed by 10 letters, followed by "n".

8.2. Is there a mailing list for translators?

Yes. Different translation groups have their own mailing lists. The list of translation projects has more information about the mailing lists and web sites run by each translation project. In addition there is freebsd-translators@freebsd.org for general translation discussion.

8.3. Are more translators needed?

Yes. The more people that work on translation the faster it gets done, and the faster changes to the English documentation are mirrored in the translated documents.

You do not have to be a professional translator to be able to help.

8.4. What languages do I need to know?

Ideally, you will have a good knowledge of written English, and obviously you will need to be fluent in the language you are translating to.

English is not strictly necessary. For example, you could do a Hungarian translation of the FAQ from the Spanish translation.

8.5. What software do I need to know?

It is strongly recommended that you maintain a local copy of the FreeBSD Git repository (at least the documentation part). This can be done by running:

```
% git clone https://git.FreeBSD.org/doc.git ~/doc
```

git.FreeBSD.org is a public git server.
This will require the git-lite package to be installed.

You should be comfortable using git. This will allow you to see what has changed between different versions of the files that make up the documentation.

For example, to view the differences between revisions `abff932fe8` and `2191c44469` of `documentation/content/en/articles/committers-guide/_index.adoc`, run:

```
% git diff abff932fe8 2191c44469 documentation/content/en/articles/committers-guide/_index.adoc
```

Please see the complete explanation of using Git in FreeBSD in the FreeBSD Handbook.

8.6. How do I find out who else might be translating to the same language?

The Documentation Project translations page lists the translation efforts that are currently known about. If others are already working on translating documentation to your language, please do not duplicate their efforts. Instead, contact them to see how you can help.

If no one is listed on that page as translating for your language, then send a message to the FreeBSD documentation project mailing list in case someone else is thinking of doing a translation, but has not announced it yet.

8.7. No one else is translating to my language. What do I do?

Congratulations, you have just started the "FreeBSD your-language-here Documentation Translation Project". Welcome aboard.

First, decide whether or not you have got the time to spare. Since you are the only person working on your language at the moment it is going to be your responsibility to publicize your work and coordinate any volunteers that might want to help you.

Write an email to the Documentation Project mailing list, announcing that you are going to translate the documentation, so the Documentation Project translations page can be maintained.

If there is already someone in your country providing FreeBSD mirroring services you should contact them and ask if you can have some webspace for your project, and possibly an email address or mailing list services.

Then pick a document and start translating. It is best to start with something fairly small - either the FAQ, or one of the tutorials.
8.8. I have translated some documentation, where do I send it?

That depends. If you are already working with a translation team (such as the Japanese team, or the German team) then they will have their own procedures for handling submitted documentation, and these will be outlined on their web pages.

If you are the only person working on a particular language (or you are responsible for a translation project and want to submit your changes back to the FreeBSD project) then you should send your translation to the FreeBSD project (see the next question).

8.9. I am the only person working on translating to this language, how do I submit my translation?

First, make sure your translation is organized properly. This means that it should drop into the existing documentation tree and build straight away.

Directories below this are named according to the language code they are written in, as defined in ISO639 (/usr/share/misc/iso639 on a version of FreeBSD newer than 20th January 1999).

⚠️ Hugo need the language codes in lowercase. For example, instead of pt_BR Hugo uses pt-br.

Currently, the FreeBSD documentation is stored in a top level directory called documentation/.
Directories below this are named according to the language code they are written in, as defined in ISO639 (/usr/share/misc/iso639 on a version of FreeBSD newer than 20th January 1999).

If your language can be encoded in different ways (for example, Chinese) then there should be directories below this, one for each encoding format you have provided.

Finally, you should have directories for each document.

For example, a hypothetical Swedish translation might look like:

documentation/
    content/
        sv/
            books/
            faq/
                _index.adoc

_sv is the name of the translation, in lang form. Note the two Makefiles, which will be used to build the documentation.

Use git diff command to generate a diff and send it to the reviews system.
You should use Bugzilla to submit a report indicating that you have submitted the documentation. It would be very helpful if you could get other people to look over your translation and double check it first, since it is unlikely that the person committing it will be fluent in the language.

Someone (probably the Documentation Project Manager, currently Documentation Engineering Team <doceng@FreeBSD.org>) will then take your translation and confirm that it builds. In particular, the following things will be looked at:

1. Does `make` in the root directory work correctly?

If there are any problems then whoever is looking at the submission will get back to you to work them out.

If there are no problems your translation will be committed as soon as possible.

### 8.10. Can I include language or country specific text in my translation?

We would prefer that you did not.

For example, suppose that you are translating the Handbook to Korean, and want to include a section about retailers in Korea in your Handbook.

There is no real reason why that information should not be in the English (or German, or Spanish, or Japanese, or …) versions as well. It is feasible that an English speaker in Korea might try to pick up a copy of FreeBSD whilst over there. It also helps increase FreeBSD’s perceived presence around the globe, which is not a bad thing.

If you have country specific information, please submit it as a change to the English Handbook (using Bugzilla) and then translate the change back to your language in the translated Handbook.

Thanks.

#### 8.10.1. Addressing the reader

In the English documents, the reader is addressed as “you”, there is no formal/informal distinction as there is in some languages.

If you are translating to a language which does distinguish, use whichever form is typically used in other technical documentation in your language. If in doubt, use a mildly polite form.

#### 8.10.2. Do I need to include any additional information in my translations?

Yes.

The header of the English version of each document will look something like this:
The exact boilerplate may change, but it will always include a "$FreeBSD$" line and the phrase "The FreeBSD Documentation Project." Note that the "$FreeBSD$" part is expanded automatically by Git, so it should be empty (just "$FreeBSD$") for new files.

Your translated documents should include their own FreeBSD line, and change the FreeBSD Documentation Project line to The FreeBSD language Documentation Project.

In addition, you should add a third line which indicates which revision of the English text this is based on.

So, the Spanish version of this file might start:

---

title: Soporte para segundos intercalares en FreeBSD
---

= Soporte para segundos intercalares en FreeBSD
Chapter 9. PO Translations

9.1. Introduction

The GNU gettext system offers translators an easy way to create and maintain translations of documents. Translatable strings are extracted from the original document into a PO (Portable Object) file. Translated versions of the strings are entered with a separate editor. The strings can be used directly or built into a complete translated version of the original document.

9.2. Quick Start

The procedure shown in Quick Start is assumed to have already been performed. The TRANSLATOR option is required and already enabled by default in the textproc/docproj port.

This example shows the creation of a Spanish translation of the short Leap Seconds article.

Procedure: Install a PO Editor

1. A PO editor is needed to edit translation files. This example uses editors/poedit.

   # pkg install poedit

Procedure: Initial Setup

When a new translation is first created, the directory structure must be created or copied from the English original:

1. Create a directory for the new translation. The English article source is in ~/doc/documentation/content/en/articles/leap-seconds/. The Spanish translation will go in ~/doc/documentation/content/es/articles/leap-seconds/. The path is the same except for the name of the language directory. The English article source is in ~/doc/en/articles/leap-seconds/. The Spanish translation will go in ~/doc/es/articles/leap-seconds/. The path is the same except for the name of the language directory.

   % mkdir ~/doc/documentation/content/es/articles/leap-seconds

2. Copy the _index.adoc from the original document into the translation directory:

   % cp ~/doc/documentation/content/en/articles/leap-seconds/_index.adoc \   ~/doc/documentation/content/es/articles/leap-seconds/
Procedure: Translation

Translating a document consists of two steps: extracting translatable strings from the original document, and entering translations for those strings. These steps are repeated until the translator feels that enough of the document has been translated to produce a usable translated document.

1. Extract the translatable strings from the original English version into a PO file:

```bash
% cd ~/doc
% po4a-gettextize \
    --format asciidoc \
    --option compat=asciidoc \
    --option yfm_keys=title,part,description \
    --master "documentation/content/en/articles/leap-seconds/_index.adoc" \
    --master-charset "UTF-8" \
    --copyright-holder "The FreeBSD Project" \
    --package-name "FreeBSD Documentation" \
    --po "documentation/content/es/articles/leap-seconds/_index.po"
```

2. Use a PO editor to enter translations in the PO file. There are several different editors available. poedit from editors/poedit is shown here.

```bash
% poedit documentation/content/es/articles/leap-seconds/_index.po
```
Procedure: Generating a Translated Document

1. Generate the translated document:

```bash
% cd ~/doc
% po4a-translate \
   --format asciidoc \ 
   --option compat=asciidoc \ 
   --option yfm_keys=title,part,description \ 
   --master "documentation/content/en/articles/leap-seconds/_index.adoc" \ 
   --master-charset "UTF-8" \ 
   --po "documentation/content/es/articles/leap-seconds/_index.po" \ 
   --localized "documentation/content/es/articles/leap-seconds/_index.adoc" \ 
   --localized-charset "UTF-8" \ 
   --keep 0
```

The name of the generated document matches the name of the English original, usually _index.adoc.

2. Check the generated file by rendering it to HTML and viewing it with a web browser:

```bash
% cd ~/doc/documentation
% make
```

9.3. Creating New Translations

The first step to creating a new translated document is locating or creating a directory to hold it. FreeBSD puts translated documents in a subdirectory named for their language and region in the format `lang.lang`. `lang` is a two-character lowercase code.

<table>
<thead>
<tr>
<th>Language</th>
<th>Region</th>
<th>Translated Directory Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>United States</td>
<td>en</td>
</tr>
<tr>
<td>Bengali</td>
<td>Bangladesh</td>
<td>bn-bd</td>
</tr>
<tr>
<td>Danish</td>
<td>Denmark</td>
<td>da</td>
</tr>
<tr>
<td>German</td>
<td>Germany</td>
<td>de</td>
</tr>
<tr>
<td>Greek</td>
<td>Greece</td>
<td>el</td>
</tr>
<tr>
<td>Spanish</td>
<td>Spain</td>
<td>es</td>
</tr>
<tr>
<td>French</td>
<td>France</td>
<td>fr</td>
</tr>
<tr>
<td>Hungarian</td>
<td>Hungary</td>
<td>hu</td>
</tr>
<tr>
<td>Italian</td>
<td>Italy</td>
<td>it</td>
</tr>
<tr>
<td>Language</td>
<td>Region</td>
<td>Translated Directory Name</td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>Japanese</td>
<td>Japan</td>
<td>ja</td>
</tr>
<tr>
<td>Korean</td>
<td>Korea</td>
<td>ko</td>
</tr>
<tr>
<td>Mongolian</td>
<td>Mongolia</td>
<td>mn</td>
</tr>
<tr>
<td>Dutch</td>
<td>Netherlands</td>
<td>nl</td>
</tr>
<tr>
<td>Polish</td>
<td>Poland</td>
<td>pl</td>
</tr>
<tr>
<td>Portuguese</td>
<td>Brazil</td>
<td>pt-br</td>
</tr>
<tr>
<td>Russian</td>
<td>Russia</td>
<td>ru</td>
</tr>
<tr>
<td>Turkish</td>
<td>Turkey</td>
<td>tr</td>
</tr>
<tr>
<td>Chinese</td>
<td>China</td>
<td>zh-cn</td>
</tr>
<tr>
<td>Chinese</td>
<td>Taiwan</td>
<td>zh-tw</td>
</tr>
</tbody>
</table>

The translations are in subdirectories of the main documentation directory, here assumed to be `~/doc/documentation/` as shown in Quick Start. For example, German translations are located in `~/doc/documentation/content/de/`, and French translations are in `~/doc/documentation/content/fr/`.

Each language directory contains separate subdirectories named for the type of documents, usually articles/ and books/.

Combining these directory names gives the complete path to an article or book. For example, the French translation of the NanoBSD article is in `~/doc/documentation/content/fr/articles/nanobsd/`, and the Mongolian translation of the Handbook is in `~/doc/documentation/content/mn/books/handbook/`.

A new language directory must be created when translating a document to a new language. If the language directory already exists, only a subdirectory in the articles/ or books/ directory is needed.
Example 11. Creating a Spanish Translation of the Porter's Handbook

Create a new Spanish translation of the Porter’s Handbook. The original is a book in ~/doc/documentation/content/en/books/porters-handbook/.

1. The Spanish language books directory ~/doc/documentation/content/es/books/ already exists, so only a new subdirectory for the Porter’s Handbook is needed:

```
% cd ~/doc/documentation/content/es/books
% mkdir porters-handbook
```

2. Copy the content from the original book:

```
% cd porters-handbook
% cp -R ~/doc/documentation/content/en/books/porters-handbook/* .
```

Now the document structure is ready for the translator to begin translating with po4a command.

9.4. Translating

The gettext system greatly reduces the number of things that must be tracked by a translator. Strings to be translated are extracted from the original document into a PO file. Then a PO editor is used to enter the translated versions of each string.

The FreeBSD PO translation system does not overwrite PO files, so the extraction step can be run at any time to update the PO file.

A PO editor is used to edit the file. editors/poedit is shown in these examples because it is simple and has minimal requirements. Other PO editors offer features to make the job of translating easier. The Ports Collection offers several of these editors, including devel/gtranslator.

It is important to preserve the PO file. It contains all of the work that translators have done.
1. Change to the base directory and update all PO files.

```bash
% cd ~/doc
% po4a-gettextize
  --format asciidoc
  --option compat=asciidoctor
  --option yfm_keys=title,part,description
  --master "documentation/content/en/books/porters-handbook/_index.adoc"
  --master-charset UTF-8
  --copyright-holder "The FreeBSD Project"
  --package-name "FreeBSD Documentation"
  --po "documentation/content/es/books/porters-handbook/_index.po"
```

2. Enter translations using a PO editor:

```bash
% poedit documentation/content/es/books/porters-handbook/_index.po
```

These steps are necessary for all .adoc files, excluding chapters-order.adoc and toc-* adoc.

### 9.5. Tips for Translators

#### 9.5.1. Preserving AsciiDoc macros

Preserve AsciiDoc macros that are shown in the English original.

**Example 13. Preserving AsciiDoc macros**

**English original:**

```asciidoc
msgid ""
"This example shows the creation of a Spanish translation of the short "
"extref:{leap-seconds}[Leap Seconds] article."
```

**Spanish translation:**

```asciidoc
msgid ""
"Este ejemplo muestra la creación de un artículo con poco contenido como el
articulo "
"extref:{leap-seconds}[Leap Seconds]."
```
9.5.2. Preserving Spaces

Preserve existing spaces at the beginning and end of strings to be translated. The translated version must have these spaces also.

9.5.3. Verbatim Tags

The contents of some tags should be copied verbatim, not translated:

- `man(1)`
- `package`
- `link`
- `image`
- `include`
- `Admonitions`
- `id’s`
- `Heading tags`
- `source`

9.6. Building a Translated Document

A translated version of the original document can be created at any time. Any untranslated portions of the original will be included in English in the resulting document. Most PO editors have an indicator that shows how much of the translation has been completed. This makes it easy for the translator to see when enough strings have been translated to make building the final document worthwhile.

9.7. Submitting the New Translation

Prepare the new translation files for submission. This includes adding the files to the version control system, setting additional properties on them, then creating a diff for submission.

The diff files created by these examples can be attached to a documentation bug report or code review.
Example 14. Spanish Translation of the NanoBSD Article

1. Create a diff of the new files from the ~/doc/ base directory so the full path is shown with the filenames. This helps committers identify the target language directory.

```bash
% cd ~/doc
% git diff documentation/content/es/articles/nanobsd/ > /tmp/es_nanobsd.diff
```
Chapter 10. Manual Pages

10.1. Introduction

Manual pages, commonly shortened to man pages, were conceived as readily-available reminders for command syntax, device driver details, or configuration file formats. They have become an extremely valuable quick-reference from the command line for users, system administrators, and programmers.

Although intended as reference material rather than tutorials, the EXAMPLES sections of manual pages often provide detailed use case.

Manual pages are generally shown interactively by the man(1) command. When the user types man ls, a search is performed for a manual page matching ls. The first matching result is displayed.

10.2. Sections

Manual pages are grouped into sections. Each section contains manual pages for a specific category of documentation:

<table>
<thead>
<tr>
<th>Section Number</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>General Commands</td>
</tr>
<tr>
<td>2</td>
<td>System Calls</td>
</tr>
<tr>
<td>3</td>
<td>Library Functions</td>
</tr>
<tr>
<td>4</td>
<td>Kernel Interfaces</td>
</tr>
<tr>
<td>5</td>
<td>File Formats</td>
</tr>
<tr>
<td>6</td>
<td>Games</td>
</tr>
<tr>
<td>7</td>
<td>Miscellaneous</td>
</tr>
<tr>
<td>8</td>
<td>System Manager</td>
</tr>
<tr>
<td>9</td>
<td>Kernel Developer</td>
</tr>
</tbody>
</table>

10.3. Markup

Various markup forms and rendering programs have been used for manual pages. FreeBSD has used groff(7) and the newer mandoc(1). Most existing FreeBSD manual pages, and all new ones, use the mdoc(7) form of markup. This is a simple line-based markup that is reasonably expressive. It is mostly semantic: parts of text are marked up for what they are, rather than for how they should appear when rendered. There is some appearance-based markup which is usually best avoided.

Manual page source is usually interpreted and displayed to the screen interactively. The source files can be ordinary text files or compressed with gzip(1) to save space.
Manual pages can also be rendered to other formats, including PostScript for printing or PDF generation. See man(1).

### 10.3.1. Manual Page Sections

Manual pages are composed of several standard sections. Each section has a title in upper case, and the sections for a particular type of manual page appear in a specific order. For a category 1 General Command manual page, the sections are:

<table>
<thead>
<tr>
<th>Section Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>Name of the command</td>
</tr>
<tr>
<td>SYNOPSIS</td>
<td>Format of options and arguments</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>Description of purpose and usage</td>
</tr>
<tr>
<td>ENVIRONMENT</td>
<td>Environment settings that affect operation</td>
</tr>
<tr>
<td>EXIT STATUS</td>
<td>Error codes returned on exit</td>
</tr>
<tr>
<td>EXAMPLES</td>
<td>Examples of usage</td>
</tr>
<tr>
<td>COMPATIBILITY</td>
<td>Compatibility with other implementations</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>Cross-reference to related manual pages</td>
</tr>
<tr>
<td>STANDARDS</td>
<td>Compatibility with standards like POSIX</td>
</tr>
<tr>
<td>HISTORY</td>
<td>History of implementation</td>
</tr>
<tr>
<td>BUGS</td>
<td>Known bugs</td>
</tr>
<tr>
<td>AUTHORS</td>
<td>People who created the command or wrote the manual page.</td>
</tr>
</tbody>
</table>

Some sections are optional, and the combination of sections for a specific type of manual page vary. Examples of the most common types are shown later in this chapter.

### 10.3.2. Macros

mdoc(7) markup is based on macros. Lines that begin with a dot contain macro commands, each two or three letters long. For example, consider this portion of the ls(1) manual page:
A Document date and Document title are defined.

A Section header for the NAME section is defined. Then the Name of the command and a one-line Name description are defined.

The SYNOPSIS section begins. This section describes the command-line options and arguments accepted.

Name (.Nm) has already been defined, and repeating it here just displays the defined value in the text.

An Optional Flag called -libxo is shown. The Fl macro adds a dash to the beginning of flags, so this appears in the manual page as --libxo.

A long list of optional single-character flags are shown.

An optional -D flag is defined. If the -D flag is given, it must be followed by an Argument. The argument is a format, a string that tells ls(1) what to display and how to display it. Details on the format string are given later in the manual page.

A final optional argument is defined. Since no name is specified for the argument, the default of file … is used.

The Section header for the DESCRIPTION section is defined.

When rendered with the command man ls, the result displayed on the screen looks like this:
NAME
ls - list directory contents

SYNOPSIS
ls [---libxo] [-ABCFGHILPRSTUWZabcdfghiklmnopqrstuvwxyz1,] [-D format]
[file ...]

DESCRIPTION
For each operand that names a file of a type other than directory, ls
displays its name as well as any requested, associated information. For
each operand that names a file of type directory, ls displays the names
of files contained within that directory, as well as any requested,
associated information.

Optional values are shown inside square brackets.

10.3.3. Markup Guidelines
The mdoc(7) markup language is not very strict. For clarity and consistency, the FreeBSD
Documentation project adds some additional style guidelines:

Only the first letter of macros is upper case
Always use upper case for the first letter of a macro and lower case for the remaining letters.

Begin new sentences on new lines
Start a new sentence on a new line, do not begin it on the same line as an existing sentence.

Update .Dd when making non-trivial changes to a manual page
The Document date informs the reader about the last time the manual page was updated. It is
important to update whenever non-trivial changes are made to the manual pages. Trivial
changes like spelling or punctuation fixes that do not affect usage can be made without updating
.Dd.

Give examples
Show the reader examples when possible. Even trivial examples are valuable, because what is
trivial to the writer is not necessarily trivial to the reader. Three examples are a good goal. A
trivial example shows the minimal requirements, a serious example shows actual use, and an in-
depth example demonstrates unusual or non-obvious functionality.

Include the BSD license
Include the BSD license on new manual pages. The preferred license is available from the
Committer’s Guide.

10.3.4. Markup Tricks
Add a space before punctuation on a line with macros. Example:
Note how the commas at the end of the `.Xr` lines have been placed after a space. The `.Xr` macro expects two parameters to follow it, the name of an external manual page, and a section number. The space separates the punctuation from the section number. Without the space, the external links would incorrectly point to section 4, or 8.,

### 10.3.5. Important Macros

Some very common macros will be shown here. For more usage examples, see `mdoc(7)`, `groff_mdoc(7)`, or search for actual use in `/usr/share/man/man*` directories. For example, to search for examples of the `.Bd Begin display` macro:

```bash
% find /usr/share/man/man* | xargs zgrep '.Bd'
```

### 10.3.5.1. Organizational Macros

Some macros are used to define logical blocks of a manual page.

<table>
<thead>
<tr>
<th>Organizational Macro</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>.Sh</code></td>
<td>Section header. Followed by the name of the section, traditionally all upper case. Think of these as chapter titles.</td>
</tr>
<tr>
<td><code>.Ss</code></td>
<td>Subsection header. Followed by the name of the subsection. Used to divide a <code>.Sh</code> section into subsections.</td>
</tr>
<tr>
<td><code>.Bl</code></td>
<td>Begin list. Start a list of items.</td>
</tr>
<tr>
<td><code>.El</code></td>
<td>End a list.</td>
</tr>
<tr>
<td><code>.Bd</code></td>
<td>Begin display. Begin a special area of text, like an indented area.</td>
</tr>
<tr>
<td><code>.Ed</code></td>
<td>End display.</td>
</tr>
</tbody>
</table>

### 10.3.5.2. Inline Macros

Many macros are used to mark up inline text.

<table>
<thead>
<tr>
<th>Inline Macro</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>.Nm</code></td>
<td>Name. Called with a name as a parameter on the first use, then used later without the parameter to display the name that has already been defined.</td>
</tr>
</tbody>
</table>
10.4. Sample Manual Page Structures

This section shows minimal desired man page contents for several common categories of manual pages.

10.4.1. Section 1 or 8 Command

The preferred basic structure for a section 1 or 8 command:

```
.Dd August 25, 2017
.Dt EXAMPLECMD 8
.Os
.Sh NAME
.Nm examplecmd
.Nd "command to demonstrate section 1 and 8 man pages"
.Sh SYNOPSIS
.Nm
.Op Fl v
.Sh DESCRIPTION
The
.Nm utility does nothing except demonstrate a trivial but complete manual page for a section 1 or 8 command.
.Sh SEE ALSO
.Xr exampleconf 5
.Sh AUTHORS
.An Firstname Lastname Aq Mt flastname@example.com
```

10.4.2. Section 4 Device Driver

The preferred basic structure for a section 4 device driver:

```
.Dd August 25, 2017
.Dt EXAMPLEDRIVER 4
.Os
.Sh NAME
.Nm exampledriver
.Nd "driver to demonstrate section 4 man pages"
.Sh SYNOPSIS
To compile this driver into the kernel, add this line to the kernel configuration file:
.Bd -ragged -offset indent
.Cd "device exampledriver"
```
To load the driver as a module at boot, add this line to
`Xr loader.conf 5`:

```conf
exampledriver_load="YES"
```

**DESCRIPTION**

The `driver` provides an opportunity to show a skeleton or template file for section 4 manual pages.

**HARDWARE**

The `driver` supports these cards from the aptly-named Nonexistent Technologies:

- **NT X149.2 (single and dual port)**
- **NT X149.8 (single port)**

**DIAGNOSTICS**

- **"flashing green light"**
  - Something bad happened.
- **"flashing red light"**
  - Something really bad happened.
- **"solid black light"**
  - Power cord is unplugged.

**SEE ALSO**

`example 8`

**HISTORY**

The `device` driver first appeared in
`Fx 49.2`

**AUTHORS**

- Firstname Lastname

  `flastname@example.com`

---

### 10.4.3. Section 5 Configuration File

The preferred basic structure for a section 5 configuration file:
10.5. Testing

Testing a new manual page can be challenging. Fortunately there are some tools that can assist in the task. Some of them, like `man(1)`, do not look in the current directory. It is a good idea to prefix the filename with `./` if the new manual page is in the current directory. An absolute path can also be used.

Use `mandoc(1)`'s linter to check for parsing errors:

```
% mandoc -T lint ./mynewmanpage.8
```

Use `textproc/igor` to proofread the manual page:

```
% igor ./mynewmanpage.8
```

Use `man(1)` to check the final result of your changes:

```
% man ./mynewmanpage.8
```

You can use `col(1)` to filter the output of `man(1)` and get rid of the backspace characters before loading the result in your favorite editor for spell checking:

```
% man ./mynewmanpage.8 | col -b | vim -R -
```

Spell-checking with fully-featured dictionaries is encouraged, and can be accomplished by using `textproc/hunspell` or `textproc/aspell` combined with `textproc/en-hunspell` or `textproc/en-aspell`, respectively. For instance:
10.6. Example Manual Pages to Use as Templates

Some manual pages are suitable as in-depth examples.

<table>
<thead>
<tr>
<th>Manual Page</th>
<th>Path to Source Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>cp(1)</td>
<td>/usr/src/bin/cp/cp.1</td>
</tr>
<tr>
<td>vt(4)</td>
<td>/usr/src/share/man/man4/vt.4</td>
</tr>
<tr>
<td>crontab(5)</td>
<td>/usr/src/usr.sbin/cron/crontab/crontab.5</td>
</tr>
<tr>
<td>gpart(8)</td>
<td>/usr/src/sbin/geom/class/part/gpart.8</td>
</tr>
</tbody>
</table>

10.7. Resources

Resources for manual page writers:

- man(1)
- mandoc(1)
- groff_mdoc(7)
- Practical UNIX Manuals: mdoc
- History of UNIX Manpages
Chapter 11. Writing Style

11.1. Tips

Technical documentation can be improved by consistent use of several principles. Most of these can be classified into three goals: be clear, be complete, and be concise. These goals can conflict with each other. Good writing consists of a balance between them.

11.1.1. Be Clear

Clarity is extremely important. The reader may be a novice, or reading the document in a second language. Strive for simple, uncomplicated text that clearly explains the concepts.

Avoid flowery or embellished speech, jokes, or colloquial expressions. Write as simply and clearly as possible. Simple text is easier to understand and translate.

Keep explanations as short, simple, and clear as possible. Avoid empty phrases like "in order to", which usually just means "to". Avoid potentially patronizing words like "basically". Avoid Latin terms like "i.e.," or "cf.", which may be unknown outside of academic or scientific groups.

Write in a formal style. Avoid addressing the reader as "you". For example, say "copy the file to /tmp" rather than "you can copy the file to /tmp".

Give clear, correct, tested examples. A trivial example is better than no example. A good example is better yet. Do not give bad examples, identifiable by apologies or sentences like "but really it should never be done that way". Bad examples are worse than no examples. Give good examples, because even when warned not to use the example as shown, the reader will usually just use the example as shown.

Avoid weasel words like "should", "might", "try", or "could". These words imply that the speaker is unsure of the facts, and create doubt in the reader.

Similarly, give instructions as imperative commands: not "you should do this", but merely "do this".

11.1.2. Be Complete

Do not make assumptions about the reader's abilities or skill level. Tell them what they need to know. Give links to other documents to provide background information without having to recreate it. Put yourself in the reader's place, anticipate the questions they will ask, and answer them.

11.1.3. Be Concise

While features should be documented completely, sometimes there is so much information that the reader cannot easily find the specific detail needed. The balance between being complete and being concise is a challenge. One approach is to have an introduction, then a "quick start" section that describes the most common situation, followed by an in-depth reference section.
11.2. Guidelines

To promote consistency between the myriad authors of the FreeBSD documentation, some guidelines have been drawn up for authors to follow.

**Use American English Spelling**

There are several variants of English, with different spellings for the same word. Where spellings differ, use the American English variant. "color", not "colour", "rationalize", not "rationalise", and so on.

The use of British English may be accepted in the case of a contributed article, however the spelling must be consistent within the whole document. The other documents such as books, web site, manual pages, etc. will have to use American English.

**Do not use contractions**

Do not use contractions. Always spell the phrase out in full. "Don't use contractions" is wrong.

Avoiding contractions makes for a more formal tone, is more precise, and is slightly easier for translators.

**Use the serial comma**

In a list of items within a paragraph, separate each item from the others with a comma. Separate the last item from the others with a comma and the word "and".

For example:

This is a list of one, two and three items.

Is this a list of three items, "one", "two", and "three", or a list of two items, "one" and "two and three"?

It is better to be explicit and include a serial comma:

This is a list of one, two, and three items.

**Avoid redundant phrases**

Do not use redundant phrases. In particular, "the command", "the file", and "man command" are often redundant.

For example, commands:

Wrong: Use the `git` command to update sources.

Right: Use `git` to update sources.

Filenames:

Wrong: ... in the filename `/etc/rc.local`...
Right: ... in /etc/rc.local...

Manual page references (the second example uses `citerefentry` with the `csh(1)` entity):

Wrong: See `man csh` for more information.

Right: See `csh(1)`.

For more information about writing style, see *Elements of Style*, by William Strunk.

### 11.3. Style Guide

To keep the source for the documentation consistent when many different people are editing it, please follow these style conventions.

### 11.4. One sentence per line

Use Semantic Line Breaks in the documentation, a technique called "one sentence per line". The idea of this technique is to help the users to write and read documentation. To get more information about this technique read the [Semantic Line Breaks](#) page.

This is an example which don't use "one sentence per line".

```
All human beings are born free and equal in dignity and rights. They are endowed with reason and conscience and should act towards one another in a spirit of brotherhood.
```

And this is an example which uses the technique.

```
All human beings are born free and equal in dignity and rights. They are endowed with reason and conscience and should act towards one another in a spirit of brotherhood.
```

### 11.4.1. Acronyms

Acronyms should be defined the first time they appear in a document, as in: "Network Time Protocol (NTP)". After the acronym has been defined, use the acronym alone unless it makes more sense contextually to use the whole term. Acronyms are usually defined only once per chapter or per document.

All acronyms should be enclosed using the ` character.

### 11.5. Special Character List

This list of special characters shows the correct syntax and the output when used in FreeBSD documentation. If a character is not on this list, ask about it on the FreeBSD documentation project mailing list.
<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Rendered</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copyright</td>
<td>(C)</td>
<td>©</td>
</tr>
<tr>
<td>Registered</td>
<td>(R)</td>
<td>®</td>
</tr>
<tr>
<td>Trademark</td>
<td>(TM)</td>
<td>™</td>
</tr>
<tr>
<td>Em dash</td>
<td>--</td>
<td>—</td>
</tr>
<tr>
<td>Ellipses</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Single right arrow</td>
<td>-&gt;</td>
<td>→</td>
</tr>
<tr>
<td>Double right arrow</td>
<td>=&gt;</td>
<td>⇒</td>
</tr>
<tr>
<td>Single left arrow</td>
<td>&lt;-</td>
<td>←</td>
</tr>
<tr>
<td>Double left arrow</td>
<td>&lt;=</td>
<td>⇐</td>
</tr>
</tbody>
</table>
Chapter 12. Editor Configuration

Adjusting your text editor configuration can make working on document files quicker and easier, and help documents conform to FDP guidelines.

12.1. Vim

Install from editors/vim, editors/vim-console, or editors/vim-tiny then follow the configuration instructions in Configuration.

12.1.1. Use

Press P to reformat paragraphs or text that has been selected in Visual mode. Press T to replace groups of eight spaces with a tab.

12.1.2. Configuration

Edit ~/.vimrc, adding these lines to the end of the file:
if has("autocmd")
    au BufNewFile,BufRead *\.sgml,*.ent,*.xsl,*.xml call Set_SGML()
    au BufNewFile,BufRead *\.[1-9] call ShowSpecial()
endif " has(autocmd)

function Set_Highlights()
    "match ExtraWhitespace /\s* \s*\|\s\+$/
    highlight default link OverLength ErrorMsg
    match OverLength /\%71v.\+/" return 0
endfunction " Set_Highlights()

function ShowSpecial()
    setlocal list listchars=tab:>>,trail:*,eol:\$
    hi def link nontext ErrorMsg
    return 0
endfunction " ShowSpecial()

function Set_SGML()
    setlocal number
    syn match sgmlSpecial "&[^;]*;"
    setlocal syntax=sgml
    setlocal filetype=xml
    setlocal shiftwidth=2
    setlocal textwidth=70
    setlocal tabstop=8
    setlocal softtabstop=2
    setlocal formatprg="fmt -p"
    setlocal autoindent
    setlocal smartindent
    " Rewrap paragraphs
    noremap P gqj
    " Replace spaces with tabs
    noremap T :s/ \t/ \t/<CR>
    call ShowSpecial()
    call Set_Highlights()
    return 0
endfunction " Set_SGML()

## 12.2. Emacs

Install from editors/emacs or editors/emacs-devel.

### 12.2.1. Validation

Emacs's nxml-mode uses compact relax NG schemas for validating XML. A compact relax NG schema for FreeBSD's extension to DocBook 5.0 is included in the documentation repository. To configure nxml-mode to validate using this schema, create ~/.emacs.d/schema/schemas.xml and add these lines to the file:
12.2.2. Automated Proofreading with Flycheck and Igor

The Flycheck package is available from Milkypostman's Emacs Lisp Package Archive (MELPA). If MELPA is not already in Emacs's packages-archives, it can be added by evaluating

```
(add-to-list 'package-archives '("melpa" . "http://stable.melpa.org/packages/") t)
```

Add the line to Emacs's initialization file (one of ~/.emacs, ~/.emacs.el, or ~/.emacs.d/init.el) to make this change permanent.

To install Flycheck, evaluate

```
(package-install 'flycheck)
```

Create a Flycheck checker for textproc/igor by evaluating

```
(flycheck-define-checker igor
 "FreeBSD Documentation Project sanity checker.
 :command ("igor" "-X" source-inplace)
 :error-parser flycheck-parse-checkstyle
 :modes (nxml-mode)
 :standard-input t)

(add-to-list 'flycheck-checkers 'igor 'append)
```

Again, add these lines to Emacs's initialization file to make the changes permanent.

12.2.3. FreeBSD Documentation Specific Settings

To apply settings specific to the FreeBSD documentation project, create .dir-locals.el in the root directory of the documentation repository and add these lines to the file:
12.3. **nano**

Install from [editors/nano](editors/nano) or [editors/nano-devel](editors/nano-devel).

### 12.3.1. Configuration

Copy the sample XML syntax highlight file to the user's home directory:

```bash
% cp /usr/local/share/nano/xml.nanorc ~/.nanorc
```

Use an editor to replace the lines in the `~/.nanorc` syntax "xml" block with these rules:

```bash
syntax "xml" "\([j]rs\)html?xml|xslt?$"
# trailing whitespace
color ,blue "\[:space:]\+$"
# multiples of eight spaces at the start a line
# (after zero or more tabs) should be a tab
color ,blue "^[TAB][ ]{8}+$"
# tabs after spaces
color ,yellow "( )+TAB"
# highlight indents that have an odd number of spaces
# lines longer than 70 characters
color ,yellow "^((\{2\})\+(TAB+))*[^ ]{1}[\{1 '\{1]"""
```

Process the file to create embedded tabs:

```bash
% perl -i'' -pe 's/TAB/\t/g' ~/.nanorc
```

### 12.3.2. Use

Specify additional helpful options when running the editor:
Users of `csh(1)` can define an alias in `~/.cshrc` to automate these options:

```bash
alias nano "nano -AKipwz -r 70 -T8"
```

After the alias is defined, the options will be added automatically:

```bash
% nano _index.adoc
```
Chapter 13. See Also

This document is deliberately not an exhaustive discussion of AsciiDoc and the FreeBSD Documentation Project. For more information about these, you are encouraged to see the following web sites.

13.1. The FreeBSD Documentation Project

- The FreeBSD Documentation Project web pages
- The FreeBSD Handbook

13.2. Hugo

- Hugo
- Hugo documentation

13.3. AsciiDoctor

- AsciiDoctor
- AsciiDoctor Documentation Portal

13.4. HTML

- The World Wide Web Consortium
- The HTML 5 specification
- CSS specification
- Sass
Appendix A: Examples

These examples are not exhaustive - they do not contain all the elements that might be desirable to use, particularly in a document’s front matter. For more examples of AsciiDoctor, examine the AsciiDoc source for this and other documents available in the Git doc repository, or available online starting at https://cgit.freebsd.org/doc/.

A.1. AsciiDoctor book
Example 15. AsciIDoctor book
Abstract

Abstract section

include::{chapters-path}preface/_index.adoc[leveloffset=+1]

include::{chapters-path}parti.adoc[lines=7..18]

include::{chapters-path}chapter-name/_index.adoc[leveloffset=+1]
---
title: An Example Article
authors:
  - author: Your name and surname
e-mail: foo@example.com
trademarks: ["general"]
---

= An Example Article
:doctype: article
:toc: macro
:toclevels: 1
:icons: font
:sectnums:
:sectnumlevels: 6
:source-highlighter: rouge
:experimental:

...

toc:[]

== My First Section

This is the first section in my article.

== My First Sub-Section

This is the first sub-section in my article.