Converting the FreeBSD Documentation to the EPUB format

A Prototypical Implementation

Benedict Reuschling
bcr@FreeBSD.org

BSDCan, May 2011

Overview

Overview

Current state of the FreeBSD Documentation Project

- Maintains various books, articles, an FAQ, the website and man pages
- SGML-based DocBook source code
- Output: XHTML, PDF, plain text, etc.
- 19 (!) translation projects
- Dedicated committers and contributors
- Good starting point for contributors into the FreeBSD project

Carrying the documentation set in your pocket?

- read it whenever and whereever you want
- show it to other people
- look something up when there is no network access
- search it using fulltext

Carrying the documentation set in your pocket?

- read it whenever and whereever you want
- show it to other people
- look something up when there is no network access
- search it using fulltext

- eBooks are becoming popular

Carrying the documentation set in your pocket?

- read it whenever and whereever you want
- show it to other people
- look something up when there is no network access
- search it using fulltext

- eBooks are becoming popular
- Widespread adoption of mobile devices
  - Different screen sizes
  - horizontal scrolling is bad
  - so is constant zooming in and out
  - → Fit the screen – any size

Overview

What is EPUB?

. . . and why we shouldn't use PDF for everything

From the EPUB specification at `idpf.org`:

> *"EPUB enables the creation and transport of reflowable digital books and other types of content as single-file digital publications that are interoperable between disparate EPUB-compliant reading devices and applications."*

What is EPUB?

. . . and why we shouldn't use PDF for everything

From the EPUB specification at `idpf.org`:

"EPUB enables the creation and transport of reflowable digital books and other types of content as single-file digital publications that are interoperable between disparate EPUB-compliant reading devices and applications."

".epub" allows publishers to produce and send a single digital publication file through distribution and offers consumers interoperability between software/hardware for unencrypted reflowable digital books and other publications.

EPUB Standards

EPUB is composed of the following three open IDPF standards:

OPF: Open Packaging Format
What does each page look like?

EPUB Standards

EPUB is composed of the following three open IDPF standards:

OPF: Open Packaging Format
What does each page look like?

OPS: Open Publication Structure
What's contained in the book?

EPUB Standards

EPUB is composed of the following three open IDPF standards:

OPF: Open Packaging Format
What does each page look like?

OPS: Open Publication Structure
What's contained in the book?

OCF: Open Container Format
What's holding it together?

Let's have a look

. . . and get our hands dirty, at last!

```
$ file example.epub
```

Let's have a look

. . . and get our hands dirty, at last!

```
$ file example.epub
example.epub: Zip archive data, at least v1.0 to extract
```

Let's have a look

. . . and get our hands dirty, at last!

```
$ file example.epub
example.epub: Zip archive data, at least v1.0 to extract
```

Could it be that simple?

Let's have a look

. . . and get our hands dirty, at last!

```
$ file example.epub
example.epub: Zip archive data, at least v1.0 to extract
```

Could it be that simple? Answer: Yes, at least this part.

eBooks: Easy to open as their dead tree counterparts

```
$ unzip example.epub
```

eBooks: Easy to open as their dead tree counterparts

```
$ unzip example.epub
Archive:  example.epub
 extracting: mimetype
  inflating: META-INF/container.xml
  inflating: OPS/chapter1.xml
  inflating: OPS/content.ncx
  inflating: OPS/content.opf
  inflating: OPS/image.png
  inflating: OPS/docbook.css
```

Speaking of trees . . .

```
.epub
├── mimetype
├── META-INF/
│   └── container.xml
└── OPS/
    ├── chapter1.xml
    ├── content.ncx
    ├── content.opf
    ├── image.png
    └── docbook.css
```

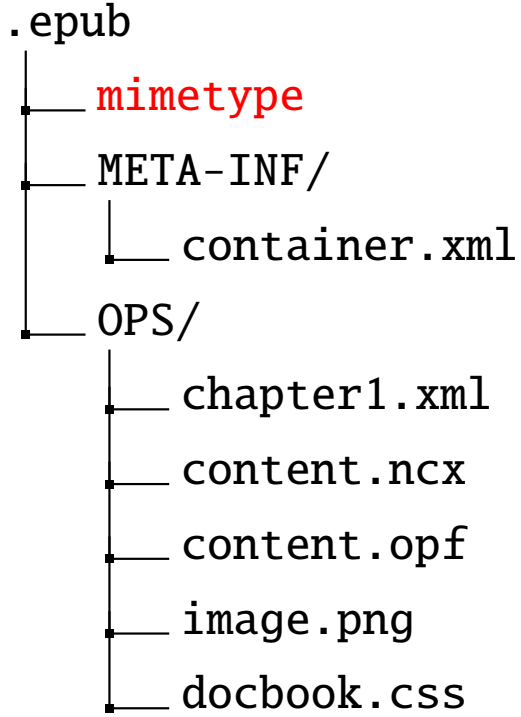
This structure from the top down to the `container.xml` is defined in the OPF, the rest can be layed out at the users convenience.

The `mimetype` file

Not much to see here:

```
$ echo "application/epub+zip" > mimetype
```

Must not be encrypted or compressed



The META-INF Directory

- Required file: `container.xml` (unencrypted)
- Describes the MIME-type and the OPS/OPF rootfile to the publication
- Optionally:

  `signatures.xml`: holds digital signatures
  `encryption.xml`: keys, algorithms, ciphers
  `rights.xml`: DRM (future versions)

```
.epub
├── mimetype
├── META-INF/
│   └── container.xml
└── OPS/
    ├── chapter1.xml
    ├── content.ncx
    ├── content.opf
    ├── image.png
    └── docbook.css
```

Example of `container.xml`

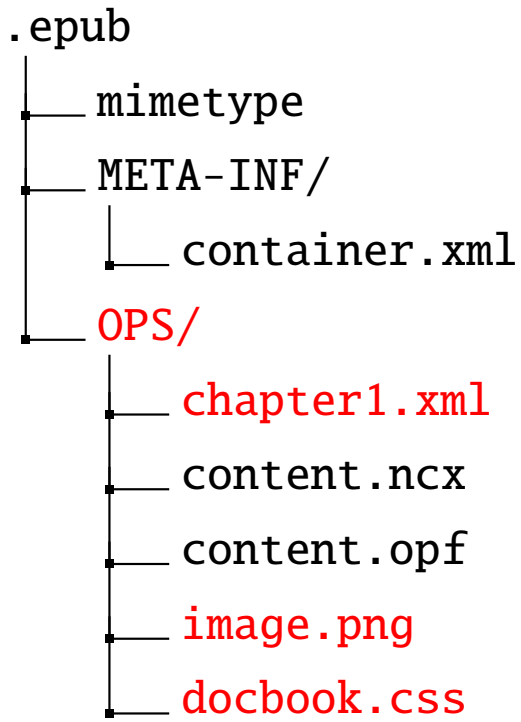
Example content

```
<?xml version="1.0"?>
<container version="1.0"
  xmlns="urn:oasis:names:tc:opendocument:xmlns:container">
  <rootfiles>
    <rootfile full-path="OPS/content.opf"
      media-type="application/oebps-package+xml" />
  </rootfiles>
</container>
```

Note: the `full-path` is *not* relative to the `META-INF` directory

Text, images and style - the book's content

- Content is XHTML 1.1 with a few exceptions
- CSS with embedded fonts to style your book
- `img` external to the book problematic
- External links work, inner references are a different story
  - Remember: it's a *reflowable* format
  - Different EPUB readers might break up pages differently
  - Rework the TOC to let users jump to specific chapters/sections
- No `script` stuff, EPUB readers are not required to execute it
- Use subdirectories to structure big books



Defining the table of contents in `content.ncx`

NCX: Navigation Center eXtended

- Borrowed from DAISY consortium
- Format to help search and navigate books for people with reading problems
- Defines the reading order using multiple `navPoints` within a `NavMap`
- Chapters, sections and subsections of the TOC are also defined here

```
.epub
├── mimetype
├── META-INF/
│   └── container.xml
└── OPS/
    ├── chapter1.xml
    ├── content.ncx
    ├── content.opf
    ├── image.png
    └── docbook.css
```

Example of a content.ncx file

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<!DOCTYPE ncx
  PUBLIC "-//NISO//DTD ncx 2005-1//EN"
  "http://www.daisy.org/z3986/2005/ncx-2005-1.dtd">
  <ncx xmlns="http://www.daisy.org/z3986/2005/ncx/"
   version="2005-1" xml:lang="en-US">
```

Example of a content.ncx file

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE ...
<ncx>
```

```
<head>
  <meta name="dc:Title" content="FreeBSD on Laptops"/>
  <meta name="dtb:uid"
   content="http://www.freebsd.org/doc/en_US.ISO8859-1/articles/
   laptop/index.html"/>
  <meta name="dtb:depth" content="1"/>
  <meta name="dtb:totalPageCount" content="0"/>
  <meta name="dtb:maxPageNumber" content="0"/>
</head>
```

Example of a content.ncx file

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE ...
<ncx>
  <head> ... </head>
```

```
<docTitle>
  <text>FreeBSD on Laptops</text>
</docTitle>
```

Example of a content.ncx file

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE ...
<ncx>
  <head> ... </head>
  <docTitle> ... </docTitle>
```

```
<navMap>
  <navPoint playOrder="1" id="id_Chapter_01">
    <navLabel>
      <text>FreeBSD on Laptops</text>
    </navLabel>
    <content src="chapter1.xml"/>
  </navPoint>
</navMap>
```

Example of a `content.ncx` file

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE ...
<ncx>
  <head> ... </head>
  <docTitle> ... </docTitle>
  <navMap> ... </navMap>
</ncx>
```

Whether `content.ncx` or `content.opf` is read first is up to the EPUB reader
→ Provide both to be on the safe side

Binding it all together using content.opf

XML file, referenced by container.xml

Purpose

- defines the location of the book's content
- references the NCX TOC-file

Basic structure

<package>: Header with namespaces and unique-identifier

<metadata>: Metadata like title, author, URI

<manifest>: the list of files that form the book's content

<spine>: defines the linear reading order of the book

```
.epub
├── mimetype
├── META-INF/
│   └── container.xml
└── OPS/
    ├── chapter1.xml
    ├── content.ncx
    ├── content.opf
    ├── image.png
    └── docbook.css
```

Example of a content.opf file

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<package
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:opf="http://www.idpf.org/2007/opf"
  xmlns="http://www.idpf.org/2007/opf"
  version="2.0" unique-identifier="BookId">
```

Example of a content.opf file

```
<?xml version="1.0" encoding="UTF-8" ?>
<package>
```

```
<metadata>
  <dc:language xsi:type="dcterms:RFC3066">en-US</dc:language>
  <dc:title>FreeBSD on Laptops</dc:title>
  <dc:identifier id="BookId">
    http://www.freebsd.org/doc/en_US.ISO8859-1/articles/
    laptop/index.html
  </dc:identifier>
</metadata>
```

Example of a content.opf file

```
<?xml version="1.0" encoding="UTF-8" ?>
<package>
  <metadata> ... </metadata>
```

```
<manifest>
  <item id="cover" href="cover.html"
    media-type="application/xhtml+xml"/>
  <item id="ncx" href="content.ncx"
    media-type="application/x-dtbncx+xml"/>
  <item id="Chapter_1" href="chapter1.xml"
    media-type="application/xhtml+xml"/>
  <item id="css" href="docbook.css" media-type="text/css"/>
</manifest>
```

Example of a content.opf file

```
<?xml version="1.0" encoding="UTF-8" ?>
<package>
  <metadata> ... </metadata>
  <manifest> ... </manifest>
```

```
<spine toc="ncx">
  <itemref idref="Chapter_1"/>
</spine>
```

Example of a `content.opf` file

```
<?xml version="1.0" encoding="UTF-8" ?>
<package>
  <metadata> ... </metadata>
  <manifest> ... </manifest>
  <spine> ... </spine>
</package>
```

- There is an optional `<guide>` section used for semantic information
- Defines things like the book's cover, table of contents, appendices

If you judge a book by the cover . . .

EPUB Cover Definition

- A separate XHTML file
- Supports text as well as GIF, JPG, PNG, SVG
- Defined in the `<guide>` section of `content.opf`
- First item in the `<spine>` with `linear="no"` attribute

If you judge a book by the cover . . .

EPUB Cover Definition

- A separate XHTML file
- Supports text as well as GIF, JPG, PNG, SVG
- Defined in the `<guide>` section of `content.opf`
- First item in the `<spine>` with `linear="no"` attribute

Building the EPUB file

Add the uncompressed `mimetype` as the first file without extra attributes:

```
$ zip -0X myepub.epub mimetype
adding: mimetype (stored 0%)
```

Building the EPUB file

Add the uncompressed `mimetype` as the first file without extra attributes:

```
$ zip -0X myepub.epub mimetype
adding: mimetype (stored 0%)
```

then, recursively add the rest without directory entries:

```
$ zip -9DurX myepub.epub *
adding: META-INF/container.xml (deflated 32%)
adding: OPS/chapter1.xml (deflated 63%)
adding: OPS/content.ncx (deflated 48%)
adding: OPS/content.opf (deflated 54%)
adding: OPS/cover.html (deflated 34%)
adding: OPS/image.png (deflated 4%)
adding: OPS/docbook.css (deflated 57%)
```

Done!

Validation

. . . you *do* validate your XML files, right? ;-)

- Finding errors early helps before scratching your head later
- `epubcheck` can validate the container and file structure
- Runs from the terminal, as Java web app or Java lib
- Available from pkgsrc as `textproc/epubcheck`[1] and `http://code.google.com/p/epubcheck/`

```
$ java -jar epubcheck.jar myepub.epub
```

[1] Please add me to the ports collection as well. :-)

Overview

Doing it mostly by hand

- XHTML output is presently the closest we have to XML
- Creating the basic EPUB structure is relatively easy
- Provide useful meta information
- Fill in content starting at the `body` tag
- Check pathnames of CSS, images, and other references
- Create the EPUB and validate it
- View it in different readers and devices
- Check that links from the TOC work correctly

Using an external converter

calibre[2] by Kovid Goyal

- eBook library management software
- Contains a converter with numerous options for many different formats

Conversion parameters (using default options)

- Input: 11.8 MB FreeBSD handbook as PDF (1045 pages)
- Output: Handbook.epub file

Results from the automatic conversion

- 3 MB epub file, 7120 pages in Apple iBooks on iPhone 4
- many duplicates and broken links in TOC

Note: this is one example, other tools might have different results

[2] This one *is* in the ports collection under `deskutils`, in pkgsrc under `misc`

Climbing up to the shoulders of giants

or: Two roads diverged in a wood . . .

gabor@ and hrs@ have been working on a conversion to XML DocBook

Climbing up to the shoulders of giants

or: Two roads diverged in a wood . . .

gabor@ and hrs@ have been working on a conversion to XML DocBook

Status

- Every book and article was updated to DocBook 4.2/XML
- Some entity issues remain, but we're getting there
- Investigating XSL-FO and Apache FOP options
- XML is more strict when it comes to missing tags
- Documentation build infrastructure must be changed as well

Climbing up to the shoulders of giants

or: Two roads diverged in a wood . . .

gabor@ and hrs@ have been working on a conversion to XML DocBook

Status

- Every book and article was updated to DocBook 4.2/XML
- Some entity issues remain, but we're getting there
- Investigating XSL-FO and Apache FOP options
- XML is more strict when it comes to missing tags
- Documentation build infrastructure must be changed as well

Kudos to them, not to me!

This will allow the use of DocBook's built-in EPUB converter

Lessons learned from the initial approaches

- Entries in the TOC need to be defined manually
- EPUB uses XHTML 1.1, doc project is still at 1.0
  - Luckily, differences aren't that big
  - Biggest change: `name` attributes are deprecated, use `id`'s
  - Not many changes required in `docbook.css` as well

Lessons learned from the initial approaches

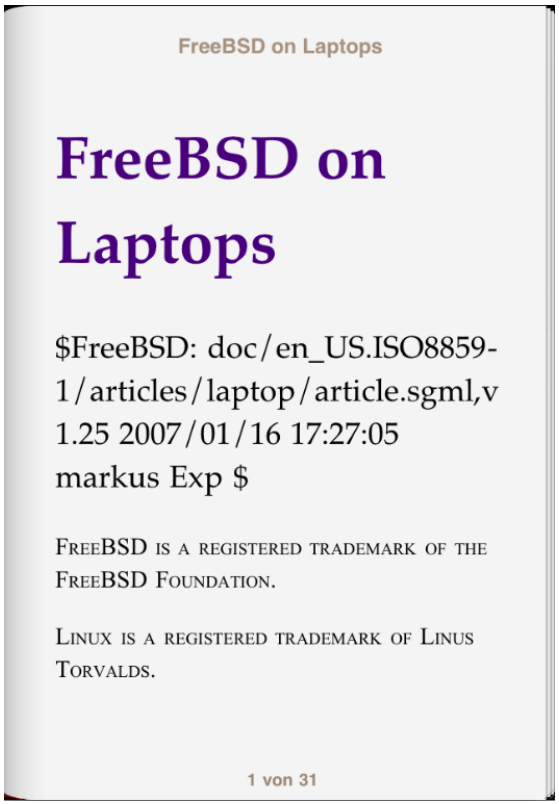
- Entries in the TOC need to be defined manually
- EPUB uses XHTML 1.1, doc project is still at 1.0
  - Luckily, differences aren't that big
  - Biggest change: `name` attributes are deprecated, use `id`'s
  - Not many changes required in `docbook.css` as well

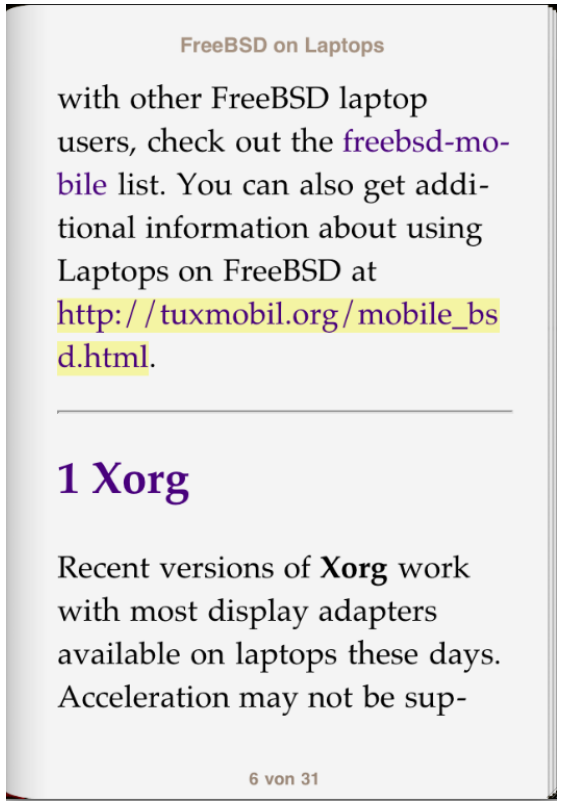
Good decisions made in the past paid out in the future

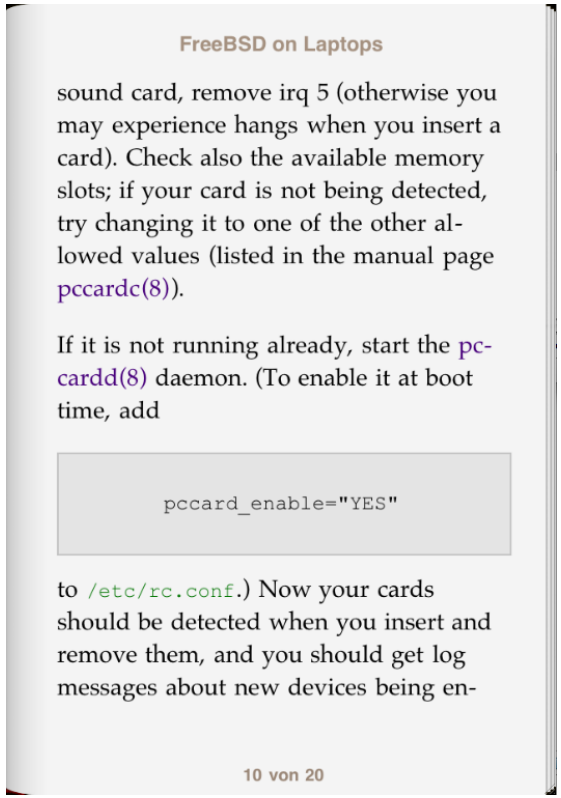
- No exotic tags in use, deprecated in newer versions
- Open standards provide reuse
- Overall documentation is a good candidate for EPUBs

→ I recommend to use the XML Docbook built-in approach

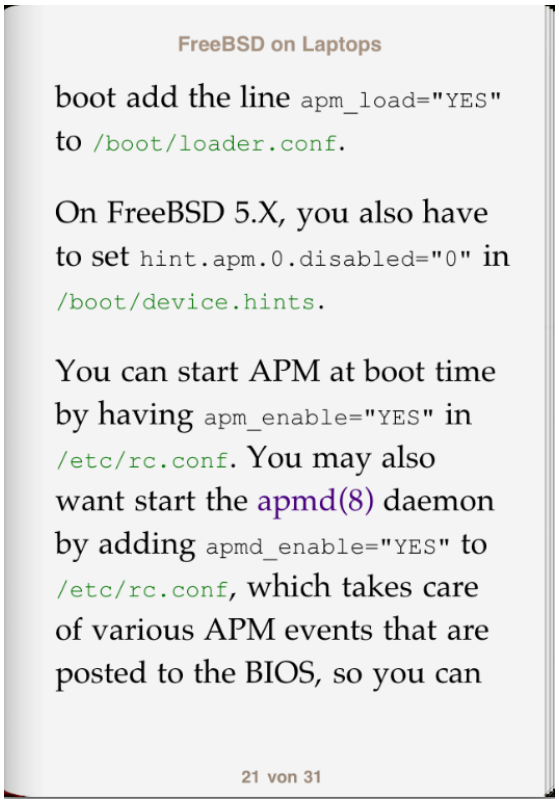
Screenshots from the laptop article in Apple iBooks

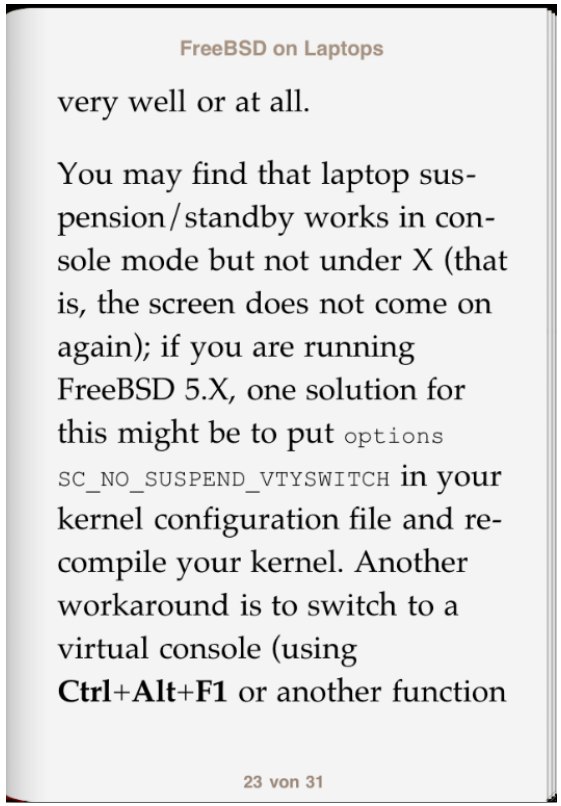


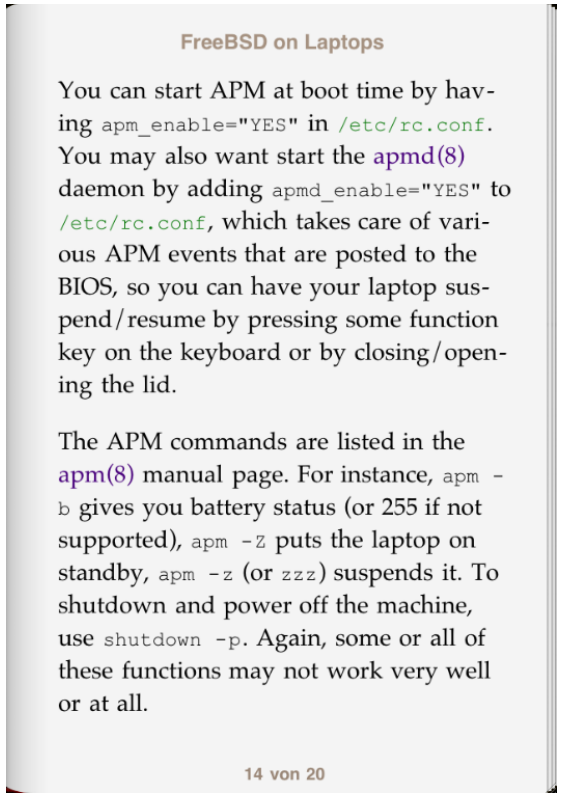




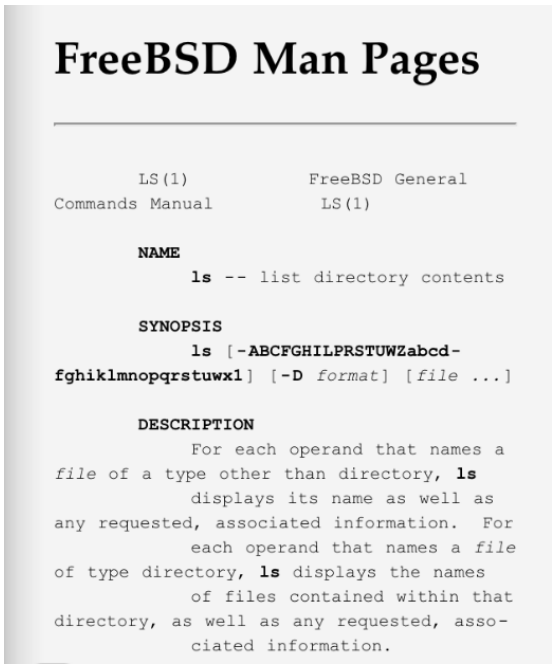
Screenshots from laptop article in Apple iBooks (2)







Screenshots from `ls(1)` in Apple iBooks



FreeBSD Man Pages

LS(1) FreeBSD General Commands Manual LS(1)

NAME

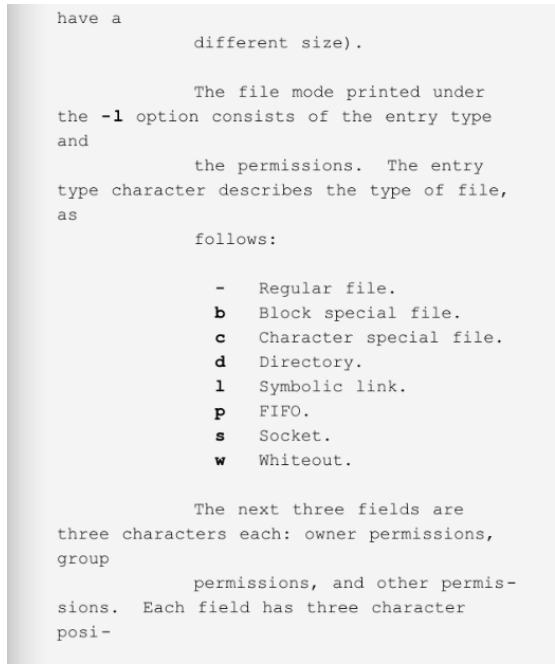
ls -- list directory contents

SYNOPSIS

ls [**-ABCFGHILPRSTUWZabcd-fghiklmnopqrstuwx1**] [**-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.

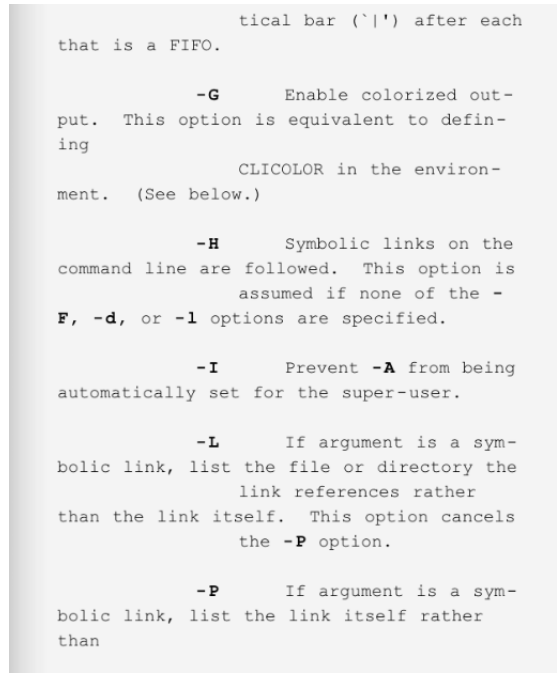


have a different size).

The file mode printed under the **-l** option consists of the entry type and the permissions. The entry type character describes the type of file, as follows:

- **-** Regular file.
- **b** Block special file.
- **c** Character special file.
- **d** Directory.
- **l** Symbolic link.
- **p** FIFO.
- **s** Socket.
- **w** Whiteout.

The next three fields are three characters each: owner permissions, group permissions, and other permissions. Each field has three character posi-



tical bar (`|') after each that is a FIFO.

-G Enable colorized output. This option is equivalent to defining CLICOLOR in the environment. (See below.)

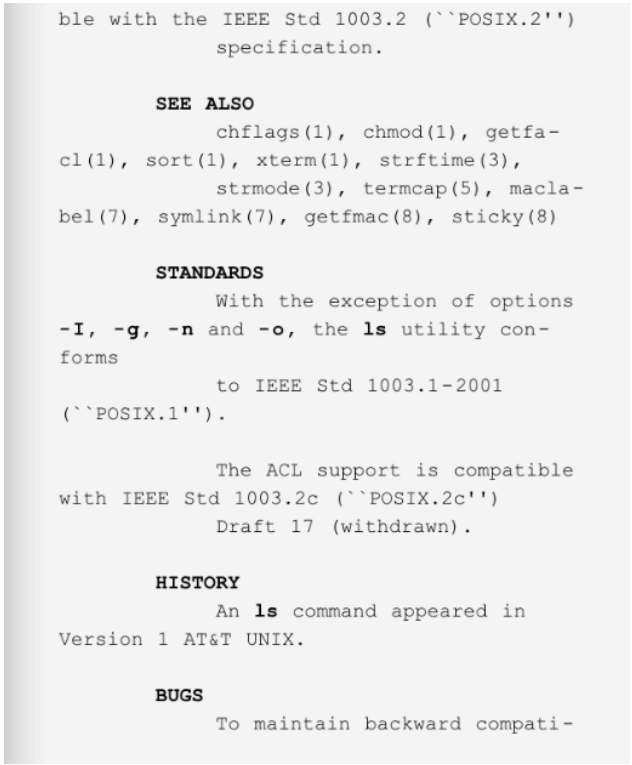
-H Symbolic links on the command line are followed. This option is assumed if none of the **-F**, **-d**, or **-l** options are specified.

-I Prevent **-A** from being automatically set for the super-user.

-L If argument is a symbolic link, list the file or directory the link references rather than the link itself. This option cancels the **-P** option.

-P If argument is a symbolic link, list the link itself rather than

Screenshots from `ls(1)` in Apple iBooks (2)



ble with the IEEE Std 1003.2 (``POSIX.2'')
specification.

SEE ALSO

chflags(1), chmod(1), getfacl(1), sort(1), xterm(1), strftime(3), strmode(3), termcap(5), maclabel(7), symlink(7), getfmac(8), sticky(8)

STANDARDS

With the exception of options **-I**, **-g**, **-n** and **-o**, the **ls** utility conforms
to IEEE Std 1003.1-2001 (``POSIX.1'').

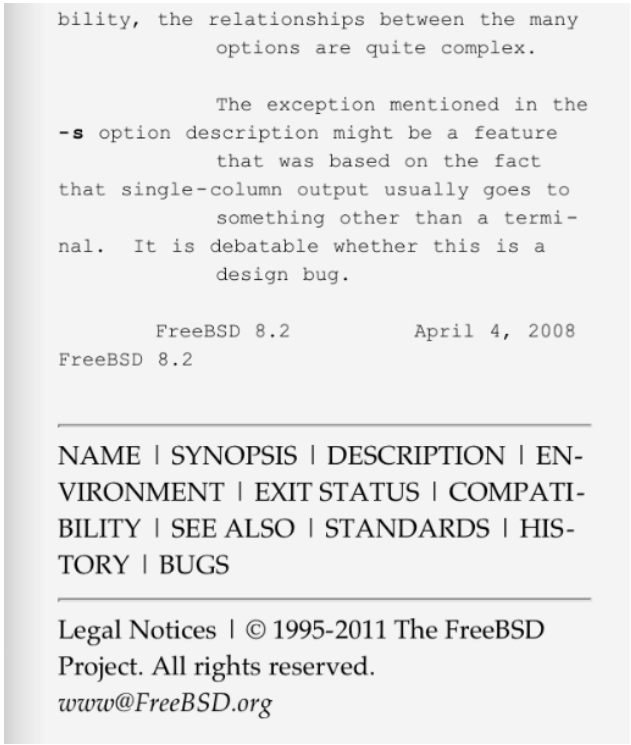
The ACL support is compatible with IEEE Std 1003.2c (``POSIX.2c'')
Draft 17 (withdrawn).

HISTORY

An **ls** command appeared in Version 1 AT&T UNIX.

BUGS

To maintain backward compati-



bility, the relationships between the many
options are quite complex.

The exception mentioned in the **-s** option description might be a feature
that was based on the fact that single-column output usually goes to
something other than a terminal. It is debatable whether this is a
design bug.

FreeBSD 8.2 April 4, 2008 FreeBSD 8.2

NAME | SYNOPSIS | DESCRIPTION | ENVIRONMENT | EXIT STATUS | COMPATIBILITY | SEE ALSO | STANDARDS | HISTORY | BUGS

Legal Notices | © 1995-2011 The FreeBSD Project. All rights reserved.
www@FreeBSD.org

Thank you for your interest and attention!

Questions, comments, feedback?

Bibliography, Links, and References

Liza Daly, IBM Developerworks

Build a digital book with EPUB - The open XML-based eBook format

http://www.ibm.com/developerworks/xml/tutorials/x-epubtut/

25 Nov 2008

International Digital Publishing Forum

EPUB specifications for OPS, OPF, OCF

http://www.openebook.org/specs.htm

09 Apr 2010

Harrison Ainsworth

Epub Format Construction Guide

http://www.hxa.name/articles/content/epub-guide_hxa7241_2007.html

27 Aug 2010