Primer do Projeto de Documentação do FreeBSD para Novos Colaboradores

Primer do Projeto de Documentação do FreeBSD para Novos Colaboradores
Revisão: 1b46489d06
2020-08-05 22:09:20 +0000 por Danilo G. Baio.

Resumo

Obrigado por fazer parte do Projeto de Documentação do FreeBSD. A sua contribuição é extremamente valiosa.

Este primer cobre tudo o que você precisa saber para começar a contribuir com o Projeto de Documentação do FreeBSD, ou FDP, incluindo ferramentas, softwares, e a filosofia por trás do Projeto de Documentação.

Este documento é um trabalho em andamento. Correções e adições são sempre bem vindas.

Índice

Lista de Tabelas

Lista de Exemplos

Prefácio

1. Prompts do Shell

Esta tabela mostra o prompt padrão do sistema e o prompt do super usuário. Os exemplos usam estes prompts para indicar com qual usuário o exemplo foi executado.

Usuário	Prompt
Usuário normal	`%`
`root`	`#`

2. Convenções Tipográficas

Esta tabela descreve as convenções tipográficas utilizadas neste livro.

Propósito	Exemplos
Nome dos comandos.	Utilize `ls -l` para listar todos os arquivos.
Nome dos arquivos.	Edite `.login`.
Saída de um programa na tela do computador.	`You have mail.`
O que o usuário digita, quando contrastado com a saída do programa na tela do computador.	`% date +"The time is %H:%M"` `The time is 09:18`
Referência a uma página de manual.	Utilize su(1) para assumir outro nome de usuário.
Nome de usuário e de grupos de usuários.	Apenas o `root` pode fazer isso.
Ênfase.	O usuário *deve* fazer isso.
Texto que o usuário deve substituir com o texto real.	Para buscar por uma palavra chave nas páginas de manual, digite `man -k` *`keyword`*
Variáveis de ambiente.	`$HOME` aponta para o diretório inicial do usuário.

3. Notas, Dicas, Informações Importantes, Avisos e Exemplos

Notas, avisos e exemplos aparecem ao longo do texto.

Nota

Notas são representadas desta forma, e contêm informações para as quais se deve ficar atento, pois podem afetar o que o usuário faz.

Dica

Dicas são representadas desta forma, e contêm informações úteis para o usuário, como as que mostram uma maneira mais fácil de fazer alguma coisa.

Importante

Informações importantes são representadas desta forma. Normalmente elas destacam passos extras que o usuário pode precisar realizar.

Atenção

Avisos são representados deste modo, e contêm informações de alerta sobre possíveis danos se não seguir as instruções. Estes danos podem ser físicos, para o equipamento ou para o usuário, ou podem ser não-físicos, tal como a deleção inadvertida de arquivos importantes.

Exemplo 1. Uma Amostra de Exemplo

Os exemplos são representados deste modo, e normalmente contêm exemplos passo a passo, ou mostram os resultados de uma determinada ação.

4. Agradecimentos

Meu muito obrigado a Sue Blake, Patrick Durusau, Jon Hamilton, Peter Flynn, e Christopher Maden, por terem gasto parte do seu tempo lendo os primeiros rascunhos deste documento e por terem oferecido muitos comentários e críticas construtivas para este trabalho.

Capítulo 1. Visão Geral

Seja bem vindo ao Projeto de Documentação do FreeBSD.(FDP). Documentação de boa qualidade é muito importante para o sucesso do FreeBSD, e nós valorizamos muito suas contribuições.

Este documento descreve como o FDP é organizado, como escrever e como submeter documentos, e como utilizar de forma efetiva as ferramentas que estão disponíveis.

Todos são bem vindos para se juntar ao FDP. A vontade de contribuir é o único requisito de adesão.

Este primer mostra como:

- Identificar quais partes do FreeBSD são mantidas pelo FDP.
- Instalar as ferramentas e arquivos de documentação necessários.
- Realizar alterações na documentação.
- Enviar de volta alterações para revisão e inclusão na documentação do FreeBSD.

1.1. Quick Start

Algumas etapas preparatórias devem ser seguidas antes de editar a documentação do FreeBSD. Primeiro, se registre na lista de email do projeto de documentação do FreeBSD. Alguns membros do time também interagem no IRC, canal `#bsddocs` na rede EFnet. Estas pessoas podem ajudar com questões e problemas envolvendo documentação.

1. Instale o meta-port textproc/docproj e o Subversion. Este meta-port instala todo software necessário para editar e compilar a documentação do FreeBSD. O pacote Subversion é necessário para obter uma cópia de trabalho da documentação e para gerar patches.

   ```
   # pkg install docproj subversion
   ```

2. Opcional: para gerar a documentação em PDF, instale o pacote textproc/fop pois ele não é instalado por padrão com o textproc/docproj.

   ```
   # pkg install fop
   ```

3. Obtenha uma cópia local da árvore de documentação do FreeBSD em `~/doc` (see Capítulo 3, *A Área de Trabalho*).

   ```
   % svn checkout https://svn.FreeBSD.org/doc/head    ~/doc
   ```

4. Configure o editor de texto:

   - Defina a quebra de linha para 70 caracteres.
   - Defina a parada de tabulação para 2.
   - Substitua cada grupo de 8 espaços por um tab.

   Configurações específicas por editor são informados em Capítulo 15, *Configuração do Editor*.

5. Atualize a árvore de trabalho local:

   ```
   % svn up ~/doc
   ```

6. Edite os arquivos de documentação que precisam de alterações. Se um arquivo precisar de grandes mudanças, consulte a lista de discussão para obter informações.

   Referencias pra uso de tag e entidade podem ser encontradas em Capítulo 8, *XHTML Markup* and Capítulo 9, *DocBook Markup*.

7. Após alteração, cheque por eventuais problemas rodando:

```
% igor -R filename.xml | less -RS
```

   Revise a saída e edite o arquivo para corrigir os problemas informados e, em seguida, execute novamente o comando para verificar os problemas restantes. Repita até que todos os erros sejam resolvidos.

8. *Sempre* realize testes de compilação antes de submeter algo. Execute **`make`** no diretório de nível superior da documentação alterada e assim será gerado a documentação no formato HTML com divisões. Por exemplo, pra compilar a versão Inglês do Handbook em HTML, execute `make` no diretório `en_US.ISO8859-1/books/handbook/` .

9. Quando as alterações estiverem completas e testadas, gere um "arquivo diff":

```
% cd ~/doc
% svn diff > bsdinstall .diff.txt
```

   Dê ao arquivo diff um nome. No exemplo acima, foram feitas alterações na parte `bsdinstall` do Handbook.

10. Submeta o arquivo diff file pela web para o sistema de Relatórios de Problema. Se estiver usando o formulário web, insira um Sumário com *[patch] `descrição curta do problema`* . Selecione o Componente `Documentation` . No campo de Descrição, insira uma breve descrição das alterações e quaisquer detalhes importantes sobre elas. Use o botão [Add an attachment] para anexar o arquivo diff. Finalmente, pressione o botão [Submit Bug] para enviar seu diff para o sistema de relatório de problemas.

1.2. Conjunto de Documentação do FreeBSD

O FDP é responsável por quatro categorias de documentação do FreeBSD.

- *Handbook*: O handbook almeja ser um compreensivo recurso de referência online para os usuários do FreeBSD.
- *FAQ*: O FAQ utiliza um formato curto de pergunta e resposta para abordar dúvidas que são frequentemente realizadas nas listas de discussão e fóruns dedicados ao FreeBSD. Este formato não permite respostas longas e detalhadas.
- *Páginas de Manual*: As páginas de manual do sistema de língua inglesa geralmente não são escritas pelo FDP, pois fazem parte do sistema base. Contudo, o FDP pode reformular partes das páginas de manual existentes para torná-las mais claras ou para corrigir imprecisões.
- *Web site*: Esta é a presença principal do FreeBSD na web, visite https://www.FreeBSD.org/ e muitos mirrors ao redor do mundo. O site é tipicamente o primeiro contato de um usuário novo com o FreeBSD.

As equipes de tradução são responsáveis por traduzir o manual e o site para diferentes idiomas. As páginas do manual não são traduzidas no momento.

Código fonte do site do FreeBSD, Handbook, e FAQ estão disponíveis no repositório de documentação em `https://svn.FreeBSD.org/doc/` .

Código fonte das páginas de manual estão disponíveis em um repositório diferente localizado em `https://svn.FreeBSD.org/base/` .

As mensagens de commit de documentação podem ser visualizadas com `svn log`. As mensagens de commit também são arquivadas em `http://lists.FreeBSD.org/mailman/listinfo/svn-doc-all` .

Endereço web para ambos os repositórios disponíveis em https://svnweb.FreeBSD.org/doc/ e https://svnweb.FreeBSD.org/base/.

Muitas pessoas tem escrito tutoriais e artigos how-to sobre FreeBSD. Alguns são armazenados como parte dos arquivos FDP. Em outros casos, o autor decidiu manter a documentação separada. O FDP esforça-se para fornecer links para o máximo possível dessas documentações externas.

Capítulo 2. Ferramentas

Várias ferramentas são utilizadas para gerenciar a documentação do FreeBSD e renderizá-la para diferentes formatos. Algumas dessas ferramentas são necessárias e devem ser instaladas antes de trabalhar com os exemplos nos capítulos a seguir. Algumas são opcionais, adicionando recursos ou tornando a tarefa de criar documentação mais simples.

2.1. Ferramentas Obrigatórias

Instale o textproc/docproj pela Coleção de Ports. Este *meta-port* instala todos os aplicativos necessários para trabalhar com a documentação do FreeBSD. Informações adicionais específicas de alguns componentes serão informadas abaixo.

2.1.1. DTDs e Entidades

A documentação do FreeBSD utiliza diversas Definições de Tipos de Documento (DTDs) e conjuntos de entidades XML. Estes são todos instalados pelo port textproc/docproj.

XHTML DTD (textproc/xhtml)
: XHTML é a linguagem markup escolhida pela Web, e é utilizada em todo o site do FreeBSD.

DocBook DTD (textproc/docbook-xml)
: O DocBook é projetado para escrever documentação técnica. A maior parte da documentação do FreeBSD é escrita no DocBook.

Entidades ISO 8879 (textproc/iso8879)
: Entidades do padrão ISO 8879:1986 utilizado por muitos DTDs. Inclui símbolos matemáticos nomeados, caracteres adicionais no conjunto de caracteres latinos (acentos, diacríticos e assim por diante) e símbolos Gregos.

2.2. Ferramentas Opcionais

Essas ferramentas não são necessárias, mas podem facilitar o trabalho na documentação ou adicionar recursos.

2.2.1. Software

Vim (editors/vim)
: Um editor de texto popular que trabalha com XML and documentos derivados, como DocBook XML.

Emacs (editors/emacs)
: Ambos editores incluem um modo especial para editar documentos marcados como XML DTD. Esse modo inclui comandos para reduzir a quantidade de digitação necessária e ajudar a reduzir a possibilidade de erros.

Capítulo 3. A Área de Trabalho

A *área de trabalho* é uma cópia da árvore de documentação do repositório do FreeBSD baixada no computador local. As alterações são feitas na cópia de trabalho local, testadas e enviadas como patches para serem submetidas no repositório principal.

Uma cópia completa da árvore de documentação pode ocupar 700 megabytes de espaço em disco. Tenha um gigabyte de espaço total para ter sobra para arquivos temporários e versões de teste dos diversos formatos de saída.

O Subversion é utilizado para gerenciar os arquivos de documentação do FreeBSD. Ele é obtido pela instalação do pacote Subversion:

```
# pkg install subversion
```

3.1. Documentação e Páginas de Manual

A documentação do FreeBSD não é formada apenas por livros e artigos. Páginas de manual para todos os comandos e arquivos de configuração também fazem parte da documentação e fazem parte do território do FDP. Dois repositórios estão envolvidos: `doc` para os livros e artigos, e `base` para o sistema operacional e páginas de manual. Para editar páginas de manual, o repositório `base` deve ser registrado separadamente.

Repositórios podem conter várias versões de documentação e código-fonte. Novas modificações quase sempre são feitas apenas para a versão mais recente, chamada `head`.

3.2. Escolhendo um Diretório

A documentação do FreeBSD é tradicionalmente armazenada em `/usr/doc/`, e o código fonte do sistema com páginas de manual em `/usr/src/`. Essas árvores são realocáveis e os usuários podem armazenar as cópias de trabalho em outros locais para evitar interferir nas informações existentes nos diretórios principais. Os exemplos a seguir utilizam `~/doc` e `~/src`, ambos subdiretórios do diretório pessoal do usuário.

3.3. Baixando uma Cópia

O processo de download de um repositório é chamado de *checkout* e é feito com um `svn checkout`. Este exemplo faz checkout de uma cópia da versão mais recente (`head`) do repositório de documentação principal:

```
% svn checkout https://svn.FreeBSD.org/doc/head    ~/doc
```

O checkout do código-fonte para trabalhar nas páginas de manual é muito semelhante:

```
% svn checkout https://svn.FreeBSD.org/base/head    ~/src
```

3.4. Atualizando

Os documentos e arquivos no repositório do FreeBSD mudam diariamente. As pessoas modificam arquivos e submetem alterações com frequência. Mesmo após um checkout inicial, já haverá alterações entre a cópia de trabalho local e o repositório principal do FreeBSD. Para atualizar a versão local com as mudanças que foram feitas no repositório principal, execute `svn update` no diretório que contém a cópia de trabalho local:

```
% svn update ~/doc
```

Adquira o hábito de proteção de executar `svn update` antes de editar os arquivos de documentação. Alguém pode ter editado esse arquivo muito recentemente e a cópia de trabalho local não incluirá essas alterações mais recentes

até que ela seja atualizada. Editar a versão mais recente de um arquivo é muito mais fácil do que tentar combinar um arquivo local editado mais antigo com a versão mais recente do repositório.

3.5. Revertendo Alterações

De vez em quando acontece que algumas mudanças não eram necessárias, ou o escritor só quer começar novamente. Arquivos podem ser "resetados" para sua forma anterior com `svn revert .` Por exemplo, para apagar as alterações feitas no `chapter.xml` e redefini-las para o formato sem modificação:

```
% svn revert chapter.xml
```

3.6. Criando um Diff

Após finalizar as alterações em um arquivo ou grupo de arquivos, as diferenças entre a cópia de trabalho local e a versão no repositório do FreeBSD devem ser coletadas em um único arquivo para ser submetido. Estes arquivos *diff* são produzidos redirecionando a saída de `svn diff` para um arquivo:

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

Dê ao arquivo um nome significativo que identifique o conteúdo. O exemplo acima é para correção ortográfica em toda a árvore de documentação.

Se o arquivo diff for enviado com a interface web "Submit a FreeBSD problem report", adicione uma extensão `.txt` para que o formulário web identifique que o conteúdo do arquivo é texto plano.

Tenha cuidado: `svn diff` inclui todas as alterações feitas no diretório atual e em quaisquer subdiretórios. Se houver arquivos na cópia de trabalho com edições que ainda não estão prontas para serem enviadas, forneça uma lista apenas dos arquivos a serem incluídos:

```
% cd ~/doc
% svn diff disks/chapter.xml printers/chapter.xml   > disks-printers.diff
```

3.7. Referências Subversion

Estes exemplos demonstram um uso muito básico do Subversion. Mais detalhes estão disponíveis no Subversion Book e na Documentação do Subversion.

Capítulo 4. Estrutura de Diretórios da Documentação

Arquivos e diretórios no repositório `doc/` seguem uma estrutura destinada a:

1. Facilitar a conversão do documento para outros formatos.
2. Promover a consistência entre as diferentes organizações de documentação, e assim facilitar a alternância entre diferentes documentos.
3. Facilitar a decisão de onde a nova documentação deve ser colocada.

Além disso, o repositório de documentação deve acomodar documentos em vários idiomas e codificações diferentes. É importante que a estrutura do repositório de documentação não imponha quaisquer padrões particulares ou preferências culturais.

4.1. O Nível Superior, doc/

Existem dois tipos de diretório em `doc/`, cada um com nomes de diretório e significados muito específicos.

Diretório	Uso
`share`	Contém arquivos que não são específicos das várias traduções e codificações da documentação. Contém subdiretórios para categorizar ainda mais as informações. Por exemplo, os arquivos que fazem parte da infra-estrutura make(1) estão em `share/mk`, enquanto o suporte adicional a arquivos XML (como o DTBook estendido do FreeBSD DTD) estão em `share/xml`.
`lang``.`*`encoding`*	Existe um diretório para cada tradução e codificação disponível da documentação, por exemplo, `en_US.ISO8859-1/` e `zh_TW.UTF-8/`. Os nomes são longos, mas ao especificar totalmente o idioma e a codificação, evitamos dores de cabeça futuras quando uma equipe de tradução desejar fornecer documentação no mesmo idioma, mas em mais de uma codificação. Isso também evita problemas que podem ser causados por uma futura mudança para Unicode.

4.2. Os Diretórios *`lang`*`.`*`encoding`*`/`

Esses diretórios contêm os próprios documentos. A documentação é dividida em até três outras categorias, indicadas pelos diferentes nomes de diretório.

Diretório	Uso
`articles`	Documentação marcada como DocBook `article` (ou equivalente). Conteúdo curto e dividido em seções. Normalmente disponível apenas como um arquivo XHTML.
`books`	Documentação marcada como DocBook `book` (ou equivalente). Conteúdo longo e dividido em capítulos.

Diretório	Uso
	Normalmente disponível como um grande arquivo XHTML (para pessoas com conexões rápidas, ou que desejam imprimi-lo facilmente de um navegador) e como uma coleção de arquivos menores e com links.
`man`	Para traduções das páginas de manual do sistema. Esse diretório conterá um ou mais diretórios `man`*N*, correspondendo às seções que foram traduzidas.

Nem todo diretório *`lang.encoding`* terá todos esses subdiretórios. Depende de quanto a tradução foi realizada por essa equipe de tradução.

4.3. Informação Específica de Documentação

Esta seção contém informações específicas sobre documentos gerenciados pelo FDP.

4.3.1. O Handbook

`books/handbook/`

O Handbook está escrito em DocBook XML usando DTD extendido do DocBook FreeBSD.

O Handbook está organizado como um DocBook `book`. O livro é dividido em `parts`, cada uma contendo vários `chapter`s. Os `chapter`s são subdivididos em seções (`sect1`) e subseções (`sect2`, `sect3`) e assim por diante.

4.3.1.1. Organização Física

Existem vários arquivos e diretórios dentro do diretório `handbook`.

Nota

A organização do Handbook pode ser alterada ao longo do tempo, e este documento pode estar defasado quanto ao detalhamento das mudanças organizacionais do mesmo. Envie perguntas sobre a organização do Handbook na lista de discussão do projeto FreeBSD.

4.3.1.1.1. `Makefile`

O `Makefile` define algumas variáveis que afetam como o fonte XML é convertido em diversos formatos e lista os vários arquivos fontes que compõem o Handbook. Em seguida, ele inclui o `doc.project.mk` padrão, para inserir o restante do código que manipula a conversão de documentos de um formato para outro.

4.3.1.1.2. `book.xml`

Este é o principal arquivo do Handbook. Ele contém a declaração DOCTYPE, bem como os elementos que descrevem a estrutura do Handbook.

`book.xml` utiliza entidades de parâmetro para carregar os arquivos com a extensão `.ent`. Esses arquivos (descritos posteriormente) e então definem as entidades gerais que são utilizadas no restante do Handbook.

4.3.1.1.3. *`directory`*`/chapter.xml`

Cada capítulo do Handbook é armazenado em um arquivo chamado `chapter.xml` em um diretório separado dos outros capítulos. Cada diretório é nomeado após o valor do atributo `id` no elemento `chapter`.

Por exemplo, se um dos arquivos de capítulo contiver:

```
<chapter id="kernelconfig">
...
</chapter>
```

Então ele será chamado de `chapter.xml` no diretório `kernelconfig`. Em geral, todo o conteúdo do capítulo está neste único arquivo.

Quando a versão XHTML do Handbook for gerada, produzirá o `kernelconfig.html`. Isso é por causa do valor `id` e não está relacionado ao nome do diretório.

Nas versões anteriores do Handbook, os arquivos eram armazenados no mesmo diretório que `book.xml` e nomeados após o valor do atributo `id` no elemento `chapter`. Agora, é possível incluir imagens em cada capítulo. Imagens para cada capítulo do Handbook são armazenadas em `share/images/books/handbook`. As imagens devem ser colocadas no mesmo diretório que os fontes XML para cada capítulo. As colisões de Namespace são inevitáveis e é mais fácil trabalhar com vários diretórios que contenham alguns arquivos, do que trabalhar com um único diretório que contenham muitos arquivos.

Com uma breve analise pode-se constatar que existem diversos diretórios com arquivos individuais `chapter.xml`, incluindo `basics/chapter.xml`, `introduction/chapter.xml`, e `printing/chapter.xml`.

Importante

Não nomeie capítulos ou diretórios com a ordenação do Handbook. Essa ordenação pode mudar conforme o conteúdo do Handbook é reorganizado. A reorganização deve ser realizada sem renomear arquivos, a menos que capítulos inteiros sejam promovidos ou rebaixados dentro da hierarquia.

Os arquivos `chapter.xml` não são arquivos completos de documentos XML que podem ser compilados individualmente. Eles só podem ser compilados como partes de todo o Handbook.

Capítulo 5. O Processo de Compilação da Documentação

Este capítulo aborda a organização do processo de compilação da documentação e como o make(1) é utilizado para isso.

5.1. Renderizando DocBook

Diferentes tipos de saída podem ser produzidos a partir de um único arquivo fonte DocBook. O tipo de saída desejado é definido com a variável `FORMATS`. Uma lista de formatos de saída conhecidos é armazenada em `KNOWN_FORMATS`:

```
% cd ~/doc/en_US.ISO8859-1/books/handbook
% make -V KNOWN_FORMATS
```

Tabela 5.1. Formatos de Saída Comuns

Valor **FORMATS**	Tipo de Arquivo	Descrição
`html`	HTML, arquivo único	Um único `book.html` ou `article.html`.
`html-split`	HTML, vários arquivos	Vários arquivos HTML, um para cada capítulo ou seção, para uso em um site comum.
`pdf`	PDF	Portable Document Format

O formato de saída padrão pode variar de acordo com o documento, mas geralmente é `html-split`. Outros formatos são escolhidos definindo `FORMATS` para um valor específico. Múltiplos formatos de saída podem ser criados de uma só vez definindo `FORMATS` para uma lista de formatos.

Exemplo 5.1. Compilar um Arquivo Único HTML

```
% cd ~/doc/en_US.ISO8859-1/books/handbook
% make FORMATS=html
```

Exemplo 5.2. Compilar HTML-Split e PDF

```
% cd ~/doc/en_US.ISO8859-1/books/handbook
% make FORMATS="html-split pdf"
```

5.2. O Conjunto de Ferramentas de Compilação da Documentação do FreeBSD

Estas são as ferramentas utilizadas para compilar e instalar a documentação do FDP.

- A principal ferramenta de compilação é o make(1), especificamente o Berkeley Make.

- A construção de pacotes é gerenciada pelo pkg-create(8) do FreeBSD.
- gzip(1) é usado para criar versões compactadas de um documento. bzip2(1) também são suportados. tar(1) é usado na criação de pacotes.
- install(1) é usado para instalar a documentação.

5.3. Noções Básicas de Makefiles no Repositório de Documentação

Existem três tipos principais de Makefile s no repositório de Documentação do Projeto FreeBSD.

- Makefile s de Subdiretório simplesmente passam os comandos para os diretórios abaixo deles.
- Makefile s de Documentação descrevem os documentos que devem ser produzidos a partir deste diretório.
- Make includes são os responsáveis pela produção do documento, e geralmente possuem o nome no formato doc.*xxx*.mk.

5.3.1. Makefiles de Subdiretório

Estes Makefile s geralmente tem a forma de:

```
SUBDIR =articles
SUBDIR+=books

COMPAT_SYMLINK = en

DOC_PREFIX?= ${.CURDIR}/..
.include "${DOC_PREFIX}/share/mk/doc.project.mk"
```

As quatro primeiras linhas não vazias definem as variáveis do make(1), SUBDIR , COMPAT_SYMLINK , e DOC_PREFIX .

A declaração SUBDIR e COMPAT_SYMLINK mostram como atribuir um valor a uma variável, sobrescrevendo qualquer valor anterior que a mesma contenha.

A segunda declaração SUBDIR mostra como um valor é anexado ao valor atual de uma variável. A variável SUBDIR agora é composta por articles books .

A declaração DOC_PREFIX mostra como um valor é atribuído para uma variável, mas somente se ela ainda não estiver definida. Isso é útil se DOC_PREFIX não for onde este Makefile pensa que é - o usuário pode cancelar e fornecer o valor correto.

Agora o que tudo isso significa? SUBDIR lista quais subdiretórios abaixo do atual devem ser incluídos no processo de compilação durante a geração do documento.

O COMPAT_SYMLINK é específico para compatibilizar os links simbólicos que ligam os idiomas a sua codificação oficial (doc/en deve apontar para en_US.ISO-8859-1).

O DOC_PREFIX é o caminho para a raíz da árvore do projeto de documentação do FreeBSD. O qual nem sempre é facil de encontrar, e que também pode ser facilmente sobrescrito, para permitir flexibilidade. O .CURDIR é uma variável interna do make(1) que contém o caminho para o diretório atual.

A linha final inclui o arquivo principal do make(1) doc.project.mk do Projeto de Documentação do FreeBSD, ele é o responsável por converter estas variáveis em instruções de compilação.

5.3.2. Makefiles de Documentação

Estes conjuntos de make(1) Makefile s descrevem como construir a documentação contida nesse diretório.

Aqui está um exemplo:

```
MAINTAINER=nik@FreeBSD.org

DOC?= book

FORMATS?= html-split html

INSTALL_COMPRESSED?= gz
INSTALL_ONLY_COMPRESSED?=

# SGML content
SRCS=  book.xml

DOC_PREFIX?= ${.CURDIR}/../../..

.include "$(DOC_PREFIX)/share/mk/docproj.docbook.mk"
```

A variável `MAINTAINER` permite que os committers reivindiquem a propriedade de um documento no Projeto de Documentação do FreeBSD, e sejam responsáveis por mantê-lo.

`DOC` é o nome (sem a extensão `.xml`) do principal documento criado por este diretório. O `SRCS` lista todos os arquivos individuais que compõem o documento. Ela também deve incluir os arquivos importantes, nos quais qualquer mudança deve resultar em uma reconstrução.

`FORMATS` indica os formatos nos quais o documento deve ser gerado por padrão. `INSTALL_COMPRESSED` contém a lista padrão das técnicas de compressão que devem ser usadas no documento depois que ele é gerado. A variável `INSTALL_ONLY_COMPRESS` , nula por padrão, deve ser definida para um valor não nulo apenas se você desejar gerar exclusivamente a versão compactada do documento.

Você já deve estar familiarizado com a atribuição da variável `DOC_PREFIX` e com as instruções de include.

5.4. Includes do Make do Projeto de Documentação do FreeBSD

make(1) includes são melhor explicados por uma inspeção de código. Aqui estão os arquivos include do sistema:

- `doc.project.mk` é o principal arquivo include do projeto, que inclui todos os arquivos includes necessários.
- `doc.subdir.mk` controla a navegação na árvore de documentação durante o processo de construção e instalação.
- `doc.install.mk` fornece as variáveis que afetam a propriedade e a instalação de documentos.
- `doc.docbook.mk` é incluído se o `DOCFORMAT` for `docbook` e se a variável `DOC` estiver definida.

5.4.1. `doc.project.mk`

Por inspeção:

```
DOCFORMAT?= docbook
MAINTAINER?= doc@FreeBSD.org

PREFIX?= /usr/local
PRI_LANG?= en_US.ISO8859-1

.if defined(DOC)
.if ${DOCFORMAT} == "docbook"
.include "doc.docbook.mk"
.endif
.endif

.include "doc.subdir.mk"
.include "doc.install.mk"
```

5.4.1.1. Variáveis

As variáveis `DOCFORMAT` e `MAINTAINER` serão atribuídas com valores padrão, se o valor das mesmas não tiver sido definido no arquivo Makefile do documento.

`PREFIX` define o caminho no qual os aplicativos de construção da documentação estão instalados. Para uma instalação normal através de pacotes e/ou ports, este caminho será sempre `/usr/local` .

A variável `PRI_LANG` deve ser configurada para refletir o idioma e a codificação nativa dos usuários aos quais os documentos se destinam. O Inglês Americano é o padrão.

Nota

A variável `PRI_LANG` de maneira alguma afeta quais documentos serão, ou que poderão, ser compilados. Sua função principal é criar links para os documentos referenciados com maior frequência no diretório raiz de instalação da documentação do FreeBSD.

5.4.1.2. Condicionais

A linha `.if defined(DOC)` é um exemplo da condicional do make (1) como em outros programas, define o comportamento se alguma condição é verdadeira ou se é falsa. `defined` é uma função que retorna se uma dada variável está definida ou não.

A seguir, `.if ${DOCFORMAT} == "docbook"` testa se a variável `DOCFORMAT` é `"docbook"` , e neste caso, inclue o `doc.docbook.mk` .

Os dois `.endif`s fecham as duas condicionais anteriores, marcando o fim da sua aplicação.

5.4.2. `doc.subdir.mk`

Este arquivo é muito longo para ser explicado em detalhes. Estas notas descrevem as principais funcionalidades.

5.4.2.1. Variáveis

- `SUBDIR` é a lista de subdiretórios nos quais o processo de construção deve ser executado.
- `ROOT_SYMLINKS` são os nomes dos diretórios que devem ser linkados para a raíz de instalação do documento a partir da sua localização atual, se o idioma atual for o idioma primário (especificado por `PRI_LANG`).
- `COMPAT_SYMLINK` já foi descrito na seção Makefiles de Subdiretório.

5.4.2.2. Targets e Macros

As dependências são descritas por *`target : dependência1 dependência2 ...`* , nas quais, para construir o `target` , é necessário primeiramente construir as dependências informadas.

Depois desta descrição, instruções de como construir o target podem ser passadas, no caso do processo de conversão entre o target e estas dependências não tiver sido previamente definido, ou se esta conversão em particular não for a mesma que a definida pelo método padrão de conversão.

A dependência especial `.USE` define o equivalente a uma macro.

```
_SUBDIRUSE: .USE
.for entry in ${SUBDIR}
 @${ECHO} "===> ${DIRPRFX}${entry}"
 @(cd ${.CURDIR}/${entry} && \
 ${MAKE} ${.TARGET:S/realpackage/package/:S/realinstall/install/} DIRPRFX=
${DIRPRFX}${entry}/ )
```

```
.endfor
```

No código acima, `_SUBDIRUSE` é agora uma macro, a qual irá executar determinados comandos quando for listada como dependência.

O que diferencia essa macro de outros targets? Basicamente, ela é executada *após* as instruções passadas no processo de construção por ser uma dependência para o mesmo, e ela não configura o `.TARGET`, que é a variável que contém o nome do target atual que está sendo construído.

```
clean: _SUBDIRUSE
 rm -f ${CLEANFILES}
```

No código acima, o `clean` usará a macro `_SUBDIRUSE` depois de ter executado a instrução `rm -f $ {CLEANFILES}`. De fato, isso faz com que `clean` vá mais a fundo na árvore de diretórios, excluindo os arquivos construídos à medida que vai *descendo* pelos subdiretórios, e não quando vai na direção oposta.

5.4.2.2.1. Targets Fornecidos

- `install` e `package` ambos percorrem a árvore de diretórios executando as suas versões reais dentro dos subdiretórios (`realinstall` e `realpackage` respectivamente).
- `clean` remove arquivos criados pelo processo de compilação (e também desce na árvore de diretórios). `cleandir` faz a mesma coisa, e também remove o diretório de objetos se este existir.

5.4.2.3. Mais Condicionais

- `exists` é outra função condicional que retorna verdadeiro se o arquivo informado existir.
- `empty` retorna verdadeiro se a variável informada estiver vazia.
- `target` retorna verdadeiro se o target informado ainda não existir.

5.4.2.4. Construções de Looping no `make (.for)`

`.for` fornece uma maneira de repetir instruções definidas para cada elemento separado por espaço em uma variável. Ele faz isso atribuíndo uma variável para conter o elemento atual da lista que está sendo examinada.

```
_SUBDIRUSE: .USE
.for entry in ${SUBDIR}
 @${ECHO} "===> ${DIRPRFX}${entry}"
 @(cd ${.CURDIR}/${entry} && \
 ${MAKE} ${.TARGET:S/realpackage/package/:S/realinstall/install/} DIRPRFX=
${DIRPRFX}${entry}/ )
.endfor
```

No código acima, se `SUBDIR` estiver vazia, nenhuma ação será executada; se ela possuir um ou mais elementos, as instruções entre `.for` e `.endfor` serão repetidas para cada elemento, com o `entry` sendo substituído com o valor do elemento atual.

Capítulo 6. O Website

O web site do FreeBSD é parte da documentação do FreeBSD. Os arquivos para o web site são armazenados no subdiretório `en_US.ISO8859-1/htdocs` do repositório `~/doc` neste exemplo.

6.1. Variáveis de Ambiente

Diversas variáveis de ambiente controlam quais partes do web site são compiladas ou instaladas e para quais diretórios.

Dica

O sistema de compilação do web site utiliza o make(1), e valida variáveis configuradas mesmo se estiverem vazias. Os exemplos aqui mostram as formas recomendadas de configurar e utilizar essas variáveis. Definir ou configurar essas variáveis com outros valores ou métodos pode levar a surpresas inesperadas.

`DOCDIR`
: DOCDIR especifica o caminho onde os arquivos do web site devem ser instalados.

  Esta variável é melhor configurada com env(1) ou o método do shell do usuário para configurar variáveis de ambiente, `setenv` para csh(1) ou `export` para sh(1).

`ENGLISH_ONLY`
: Padrão: indefinido. Compile e inclua todas as traduções.

  `ENGLISH_ONLY=yes` : compile apenas os documentos em Inglês e ignore todas as traduções.

`WEB_ONLY`
: Padrão: indefinido. Compile o web site e todos os livros e artigos.

  `WEB_ONLY=yes` : Compile ou instale apenas páginas HTML do diretório `en_US.ISO8859-1/htdocs` . Outros diretórios e documentos, incluindo livros e artigos, serão ignorados.

`WEB_LANG`
: Padrão: indefinido. Compile e inclua todos os idiomas disponíveis no web site.

  Defina com uma lista separada por espaços, todos os idiomas a serem incluídos na compilação ou instalação. Os formatos são os mesmos que os nomes de diretório no diretório raiz do documento. Por exemplo, para incluir os documentos alemão e francês:

  ```
  WEB_LANG="de_DE.ISO8859-1 fr_FR.ISO8859-1"
  ```

`WEB_ONLY` , `WEB_LANG` , e `ENGLISH_ONLY` são variáveis make(1) quee podem ser definidas em `/etc/make.conf` , `Makefile.inc` , como variáveis de ambiente na linha de comando, ou em arquivos dot.

6.2. Compilando e Instalando as Páginas Web

Após obter os arquivos fontes da documentação e web site, o site pode ser compilado.

Uma instalação real do web site precisa ser executada pelo usuário `root` porque as permissões no diretório do servidor web não permitirão a instalação de arquivos por um usuário não privilegiado. Para testar, pode ser útil instalar os arquivos com um usuário normal em um diretório temporário.

Nestes exemplos, os arquivos do web site são criados pelo usuário `jru` em seu diretório home, `~/doc`, com um caminho completo de `/usr/home/jru/doc` .

Dica

A compilação do web site utiliza o arquivo `INDEX` da Coleção de Ports e pode falhar se este arquivo ou `/usr/ports` não estiver presente no sistema. A abordagem mais simples é instalar a Coleção de Ports.

Exemplo 6.1. Compile o Web Site Completo e Todos Documentos

Compile o web site e todos os documentos. Os arquivos finais são deixados na árvore de documento:

```
% cd ~/doc/en_US.ISO8859-1/htdocs/
% make all
```

Exemplo 6.2. Compile Apenas o Web Site em Inglês

Compile o web site apenas em Inglês, como usuário `jru`, e instale os arquivos finais em `/tmp/www` para teste:

```
% cd ~/doc/en_US.ISO8859-1/htdocs/
% env DOCDIR=/tmp/www make ENGLISH_ONLY=yes WEB_ONLY=yes all install
```

Alterações em arquivos estáticos geralmente podem ser testadas visualizando os arquivos modificados diretamente com um navegador web. Se o web site foi construído como apresentado acima, a página principal modificada pode ser visualizada com:

```
% firefox /tmp/www/data/index.html
```

Modificações em arquivos dinâmicos podem ser testadas com um servidor web rodando no sistema local. Depois de construir o site como apresentado acima, o `/usr/local/etc/apache24/httpd.conf` pode ser usado com www/apache24:

```
# httpd.conf for testing the FreeBSD website
Define TestRoot "/tmp/www/data"

# directory for configuration files
ServerRoot "/usr/local"

Listen 80

# minimum required modules
LoadModule authz_core_module libexec/apache24/mod_authz_core.so
LoadModule mime_module libexec/apache24/mod_mime.so
LoadModule unixd_module libexec/apache24/mod_unixd.so
LoadModule cgi_module libexec/apache24/mod_cgi.so
LoadModule dir_module libexec/apache24/mod_dir.so

# run the webserver as user and group
User www
Group www

ServerAdmin you@example.com
ServerName fbsdtest
```

```
# deny access to all files
<Directory />
    AllowOverride none
    Require all denied
</Directory>

# allow access to the website directory
DocumentRoot "${TestRoot}"
<Directory "${TestRoot}">
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>

# prevent access to .htaccess and .htpasswd files
<Files ".ht*">
    Require all denied
</Files>

ErrorLog "/var/log/httpd-error.log"
LogLevel warn

# set up the CGI script directory
<Directory "${TestRoot}/cgi">
    AllowOverride None
    Options None
    Require all granted
    Options +ExecCGI
    AddHandler cgi-script .cgi
</Directory>

Include etc/apache24/Includes/*.conf
```

Inicie o servidor web com

```
# service apache24 onestart
```

O web site pode ser visualizado em http://localhost. Esteja ciente de que muitos links se referem ao site real do FreeBSD por nome, e esses links ainda levar para o site externo em vez da versão de teste local. O teste completo do web site local exigirá a configuração temporária do DNS para que o endereço `www.FreeBSD.org` seja resolvido como `localhost` ou o endereço IP local.

Exemplo 6.3. Compile e Instale o Web Site

Compile o web site e todos os documentos como usuário `jru`. Instale os arquivos finais como `root` no diretório padrão, `/root/public_html` :

```
% cd ~/doc/en_US.ISO8859-1/htdocs
% make all
% su -
Password:
# cd /usr/home/jru/doc/en_US.ISO8859-1/htdocs
# make install
```

O processo de instalação não exclui nenhum arquivo antigo ou desatualizado que existia anteriormente no mesmo diretório. Se uma nova cópia do web site for criada e instalada todos os dias, esse comando localizará e excluirá todos os arquivos que não foram atualizados em três dias:

```
# find /usr/local/www  -ctime 3 -delete
```

Capítulo 7. Primer XML

A maioria das documentações do FDP é escrita com linguagens markup baseadas em XML. Este capítulo explica o que isso significa, como ler e entender os arquivos fontes da documentação e as técnicas de XML utilizadas.

Partes desta seção foram inspiradas por Mark Galassi's Get Going With DocBook.

7.1. Visão Geral

Nos primórdios da era computacional, o texto eletrônico era simples. Havia poucos conjuntos de caracteres como ASCII ou EBCDIC, e apenas isso. Texto era texto, e o que você lia era realmente o texto que você tinha. Sem frescuras, sem formatação, sem inteligência.

Inevitavelmente, isso não era suficiente. Quando o texto está em um formato utilizável por computadores, espera-se que eles possam usá-lo e manipulá-lo de maneira inteligente. Os autores querem indicar que certas frases devem ser enfatizadas, adicionadas a um glossário ou transformadas em hiperlinks. Os nomes dos arquivos podem ser apresentados em uma fonte de estilo "typewriter" para exibição na tela do computador, ou como "itálico" quando impressos, ou qualquer outra opção dentre uma infinidade de opções para apresentação.

Esperava-se que a Inteligência Artificial (IA) facilitasse isso. O computador leria o documento e identificaria automaticamente frases-chave, nomes de arquivos, textos que o leitor deveria digitar, exemplos e outros tipos. Infelizmente, na vida real não foi dessa forma, e os computadores ainda precisam de assistência antes que possam processar o texto de maneira significativa.

Mais precisamente, eles precisam de ajuda para identificar o que é o quê. Considere este texto:

> Para remover `/tmp/foo` , use rm(1).
>
> ```
> % rm /tmp/foo
> ```

É fácil identificar quais partes são nomes de arquivos, quais são comandos a serem digitados, quais partes são referências a páginas de manual e assim por diante. Mas o computador que processa o documento não consegue. Para isso, precisamos utilizar markup.

"Markup" é geralmate utilizado assim "adicionando valor" ou "aumentando o custo". O termo tem seus significados realçados quando aplicado ao texto. Markup é um texto adicional incluído no documento, diferenciado de alguma forma do conteúdo do documento, para que os programas que processam o documento possam ler a marcação e utiliza-la ao tomar decisões sobre o documento. Os editores podem ocultar o markup do usuário, para que este não se distraia com ela.

A informação extras armazenada no markup *adiciona valor* ao documento. Adicionar markup ao documento normalmente deve ser feito por uma pessoa - afinal, se os computadores pudessem reconhecer o texto suficientemente bem para adicionar a markup, não haveria necessidade de utilizar markup. Isto *aumenta o custo* (o esforço necessário) para criar algum documento.

O exemplo anterior é representado neste documento da seguinte forma:

```
<para>Para remover <filename>/tmp/foo</filename> , use &man.rm.1;.</para>

<screen>&prompt.user; <userinput>rm /tmp/foo</userinput></screen>
```

O markup é claramente separado do conteúdo.

As linguagens markup definem o que as marcações significam e como elas devem ser interpretadas.

Claro, uma linguagem markup pode não ser suficiente. Uma linguagem markup para documentação técnica tem requisitos muito diferentes de uma linguagem markup destinada a receitas de culinária. Isso, por sua vez, seria

muito diferente de uma linguagem markup utilizada para descrever uma poesia. O que é realmente necessário é uma primeira linguagem utilizada para escrever essas outras linguagens markup. Uma *meta linguagem markup*.

É exatamente isso que a eXtensible Markup Language (XML) é. Muitas linguagens markup foram escritas em XML, incluindo as duas mais utilizadas pelo FDP, XHTML e DocBook.

Cada definição de idioma é mais apropriadamente chamada de gramática, vocabulário, esquema ou Definição de Tipo de Documento (DTD). Existem vários idiomas para especificar uma gramática XML ou um *schema*.

Um schema é uma especificação *completa* de todos os elementos que podem ser utilizados, a ordem em que devem aparecer, quais elementos são obrigatórios, quais são opcionais e assim por diante. Isso torna possível escrever um XML *parser* que lê tanto o schema quanto um documento que afirma estar em conformidade com o schema. O parser pode confirmar se todos os elementos exigidos pelo vocabulário estão ou não na ordem correta do documento ou se há algum erro no markup. Isto é normalmente conhecido como a "valiidação do documento".

Nota

A validação confirma se a escolha dos elementos, sua ordenação e assim por diante estão em conformidade com os listados na gramática. Ela *não* valida se o markup *correto* foi utilizado no conteúdo. Se todos os nomes de arquivo em um documento fossem marcados como sendo nomes de função, o analisador não sinalizaria isso como um erro (supondo, é claro, que o schema define elementos para nomes de arquivos e funções e que eles possam aparecer no mesmo local).

A maioria das contribuições no Projeto de Documentação utilizará markup XHTML ou DocBook, em vez de alterações nos schemas. Por esse motivo, este livro não abordará como escrever um vocabulário.

7.2. Elementos, Tags e Atributos

Todos os vocabulários escritos em XML compartilham certas características. Isso não surpreende, pois a filosofia por trás do XML inevitavelmente irá transparecer. Uma das manifestações mais óbvias desta filosofia é a do *conteúdo* e dos *elementos*.

A documentação, seja uma única página web ou um livro extenso, é considerada como conteúdo. Este conteúdo é então dividido e subdividido em elementos. A finalidade de adicionar markup é nomear e identificar os limites desses elementos para processamento futuro.

Por exemplo, considere um livro típico. No maior nível, o livro é um elemento. Este elemento "livro" contém obviamente capítulos, que podem ser considerados elementos também. Cada capítulo conterá mais elementos, como parágrafos, citações e notas de rodapé. Cada parágrafo pode conter outros elementos, identificando o conteúdo que foi um discurso direto ou o nome de um personagem na história.

Pode ser útil pensar nisso como um conteúdo por "pedaços". No nível mais alto é um pedaço, o livro. Olhando um pouco mais, encontra-se mais pedaços, os capítulos individuais. Estes são segmentados em parágrafos, notas de rodapé, nomes de caracteres e assim por diante.

Observe como essa diferenciação entre diferentes elementos do conteúdo pode ser feita sem recorrer a quaisquer termos XML. É realmente surpreendentemente simples. Isso pode ser feito com uma caneta marca-texto e um livro impresso, usando cores diferentes para indicar diferentes partes do conteúdo.

É claro que não temos um marca-texto eletrônico, então precisamos de outra maneira de indicar a qual elemento cada parte do conteúdo pertence. Em idiomas escritos em XML (XHTML, DocBook, e outros) isto é feito por meio de *tags*.

Uma tag é usada para identificar onde um determinado elemento começa e onde o elemento termina. *A tag não faz parte do próprio elemento*. Como cada gramática foi normalmente escrita para marcar tipos específicos de informação, cada um reconhecerá elementos diferentes e, portanto, terá nomes diferentes para as tags.

Para um elemento chamado *`nome-do-elemento`* , a tag inicial normalmente se parecerá com *`<nome-do-elemento >`*. A tag de fechamento correspondente para este elemento é *`</nome-do-elemento >`*.

Exemplo 7.1. Utilizando um Elemento (Tag Inicial e Final)

XHTML possui um elemento para indicar que o conteúdo incluído pelo elemento é um parágrafo, chamado `p`.

```
<p>This is a paragraph.  It starts with the start tag for
  the 'p' element, and it will end with the end tag for the 'p'
  element.</p>

<p>This is another paragraph.  But this one is much shorter.</p>
```

Alguns elementos não possuem conteúdo. Por exemplo, em XHTML, uma linha horizontal pode ser incluída no documento. Para estes elementos "vazios", XML trouxe um formato abreviado que é completamente equivalente à versão de duas tags:

Exemplo 7.2. Usando um Elemento Sem Conteúdo

XHTML tem um elemento para indicar uma linha horizontal, chamada `hr`. Esse elemento não possui conteúdo, e se parece com isso:

```
<p>One paragraph.</p>
<hr></hr>

<p>This is another paragraph.  A horizontal rule separates this
  from the previous paragraph.</p>
```

A versão abreviada consiste em uma única tag:

```
<p>One paragraph.</p>
<hr/>

<p>This is another paragraph.  A horizontal rule separates this
  from the previous paragraph.</p>
```

Como mostrado acima, os elementos podem conter outros elementos. No exemplo do livro anterior, o elemento livro continha elementos de capítulo, que por sua vez continham elementos de parágrafo, e assim por diante.

Exemplo 7.3. Elementos Dentro de Elementos; `em`

```
<p>This is a simple <em>paragraph</em> where some
  of the <em>words</em> have been <em>emphasized</em>.</p>
```

A gramática consiste em regras que descrevem quais elementos podem conter outros elementos e exatamente o que eles podem conter.

Importante

As pessoas geralmente confundem os termos tags e elementos e usam os termos como se fossem intercambiáveis. Eles não são.

Um elemento é uma parte conceitual do seu documento. Um elemento tem início e fim definidos. As tags marcam onde o elemento começa e termina.

Quando este documento (ou qualquer pessoa com conhecimento sobre XML) refere-se a "a <p> tag" significa o texto literal que consiste nos três caracteres <, p, e >. Mas a frase "o elemento p" refere-se ao elemento inteiro.

Essa distinção *é* muito sutil. Mas tenha isso em mente.

Elementos podem ter atributos. Um atributo tem um nome e um valor e é usado para adicionar informações extras ao elemento. Isso pode ser uma informação que indica como o conteúdo deve ser renderizado ou pode ser algo que identifica exclusivamente essa ocorrência do elemento ou isso pode ser outra coisa também.

Os atributos de um elemento são escritos *dentro* da tag de início para aquele elemento, e toma o formato *`nome-do-atributo`*`="`*`valor-do-atributo`*`"`.

Em XHTML, o elemento `p` tem um atributo chamado `align`, que sugere um alinhamento (justificação) do parágrafo para o programa exibindo o XHTML.

O atributo `align` pode ter um dos quatro valores definidos, `left`, `center`, `right` e `justify`. Se o atributo não for especificado, o padrão será `left`.

Exemplo 7.4. Usando um Elemento com um Atributo

```
<p align="left"> The inclusion of the align attribute
  on this paragraph was superfluous, since the default is left.</p>

<p align="center"> This may appear in the center.</p>
```

Alguns atributos só aceitam valores específicos, como `left` ou `justify`. Outros permitem qualquer valor.

Exemplo 7.5. Aspas Simples nos Atributos

```
<p align='right'> I am on the right!</p>
```

Os valores de atributos em XML devem ser colocados entre aspas simples ou duplas. Aspas duplas são tradicionais. Aspas simples são úteis quando o valor do atributo contém aspas duplas.

Informações sobre atributos, elementos e tags são armazenadas em arquivos de catálogo. O Projeto de Documentação usa catálogos padrão do DocBook e inclui catálogos adicionais para recursos específicos do FreeBSD.

Os caminhos para os arquivos de catálogo são definidos em uma variável de ambiente para que possam ser encontradas pelas ferramentas de compilação de documentos.

7.2.1. Para Fazer...

Antes de rodar os exemplos deste documento, instale o textproc/docproj pela Coleção de Ports do FreeBSD. Este é um *meta-port* que baixa e instala os programas padrão e arquivos de suporte necessários para o Projeto de Documentação. Os usuários de csh(1) devem executar o `rehash` para que o shell reconheça os novos binários depois de instalados ou efetue logout e, em seguida, faça login novamente.

1. Crie `example.xml` e insira este texto:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>An Example XHTML File</title>
  </head>

  <body>
    <p>This is a paragraph containing some text.</p>

    <p>This paragraph contains some more text.</p>

    <p align="right"> This paragraph might be right-justified.</p>
  </body>
</html>
```

2. Tente validar esse arquivo usando um parser XML.

   O textproc/docproj inclui o parser [22] `xmllint`.

   Execute `xmllint` para validar o documento:

```
% xmllint --valid --noout example.xml
```

   `xmllint` não retorna nada se o documento for validado com sucesso.

3. Veja o que acontece quando os elementos obrigatórios são omitidos. Exclua a linha com as tags `<title>` e `</title>`, então execute novamente a validação.

```
% xmllint --valid --noout example.xml
example.xml:5: element head: validity error : Element head content does not follow
 the DTD, expecting ((script | style | meta | link | object | isindex)* , ((title ,
 (script | style | meta | link | object | isindex)* , (base , (script | style | meta
 | link | object | isindex)*)?) | (base , (script | style | meta | link | object |
 isindex)* , title , (script | style | meta | link | object | isindex)*))), got ()
```

   Isso mostra que o erro de validação vem da linha *`cinco`* do arquivo *`example.xml`* e que o conteúdo de `<head>` é a parte que não segue as regras da gramática XHTML.

   Em seguida, o `xmllint` mostra a linha onde o erro foi encontrado e marca a posição exata com um sinal ^.

4. Substitua o elemento `title`.

7.3. A Declaração DOCTYPE

No início de cada documento pode-se especificar o nome do DTD ao qual o documento está em conformidade. Esta declaração DOCTYPE é usada por XML parsers para identificar o DTD e garantir que o documento esteja de acordo com ele.

Uma declaração típica para um documento escrito em conformidade com a versão 1.0 do XHTML DTD se parece com isto:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
```

Essa linha contém vários componentes diferentes.

`<!`
: O *indicador* mostra que esta é uma declaração XML.

`DOCTYPE`
: Mostra que esta é uma declaração XML do tipo de documento.

`html`
: Nomeia o primeiro elemento que aparecerá no documento.

`PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"`
: Lista o Identificador Público Formal (FPI) para o DTD com o qual este documento está em conformidade. O XML parser usa isso para encontrar o DTD correto ao processar este documento.

  `PUBLIC` não faz parte do FPI, mas indica ao XML parser como encontrar o DTD mencionado no FPI. Outras formas de informar ao XML parser como encontrar o DTD serão informadas depois.

`"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"`
: Um nome de arquivo local ou uma URL para encontrar o DTD.

`>`
: Termina a declaração e retorna ao documento.

7.3.1. Identificadores Públicos Formais (FPIs)

Nota

Não é necessário conhecer isso, mas pode ser útil e ajudar a depurar problemas quando o XML parser não conseguir localizar o DTD.

FPIs devem seguir uma sintaxe específica:

```
"Proprietário
//Palavra-chave
Descrição
//Idioma"
```

`Proprietário`
: O proprietário do FPI.

  O início da string identifica o proprietário do FPI. Por exemplo, o FPI `"ISO 8879:1986//ENTITIES Greek Symbols//EN"` lista `ISO 8879:1986` como sendo o proprietário para o conjunto de entidades de Símbolos Gregos. ISO 8879:1986 é o número na International Organization for Standardization (ISO) para o padrão SGML, o predecessor (e um superset) do XML.

  Caso contrário, essa sequência seria parecida com `-//`*`Proprietário`* ou `+//`*`Proprietário`* (observe que a única diferença é o + ou -).

  Se a string começar com -, a informação do proprietário não é registrada, com o + identifica como registrada.

A ISO 9070:1991 define como os nomes registrados são gerados. Pode ser derivado do número de uma publicação ISO, um código ISBN ou um código de organização atribuído de acordo com a ISO 6523. Além disso, uma autoridade de registro poderia ser criada para atribuir nomes registrados. O conselho da ISO delegou isso ao American National Standards Institute (ANSI).

Como o Projeto FreeBSD não foi registrado, a string de propriedade é `-//FreeBSD`. Como visto no exemplo, a W3C também não é registrada.

`Palavra-chave`
Existem várias palavras-chave que indicam o tipo de informação no arquivo. Algumas das palavras-chave mais comuns são `DTD`, `ELEMENT`, `ENTITIES` e `TEXT`. `DTD` é usada apenas para arquivos DTD, `ELEMENT` é normalmente usada para fragmentos DTD que contêm apenas declarações de elementos ou entidades. `TEXT` é usada para conteúdo XML (texto e tags).

`Descrição`
Qualquer descrição pode ser informada no conteúdo deste campo. Isso pode incluir números de versão ou qualquer texto curto que tenha significado e seja exclusivo para o sistema XML.

`Idioma`
Um código de dois caracteres ISO que identifica o idioma nativo do arquivo. `EN` é usado para o Inglês.

7.3.1.1. Arquivos `catalog`

Com a sintaxe acima, um XML parser precisa ter alguma forma de transformar o FPI no nome do arquivo que contém o DTD. Um arquivo de catálogo (normalmente chamado de `catalog`) contém linhas que mapeiam FPIs para nomes de arquivos. Por exemplo, se o arquivo de catálogo continha a linha:

```
PUBLIC  "-//W3C//DTD XHTML 1.0 Transitional//EN"              "1.0/transitional.dtd"
```

O XML parser sabe que o DTD é chamado de `transitional.dtd` no subdiretório `1.0` do diretório que continha `catalog`.

Examine o conteúdo de `/usr/local/share/xml/dtd/xhtml/catalog.xml`. Este é o arquivo de catálogo dos XHTML DTD s que foram instalados como parte do port textproc/docproj.

7.3.2. Alternativas para FPI s

Em vez de usar uma FPI para indicar o DTD ao qual o documento está em conformidade (e, portanto, qual arquivo no sistema contém o DTD), o arquivo pode ser explicitamente especificado.

A sintaxe é um pouco diferente:

```
<!DOCTYPE html SYSTEM "/path/to/file.dtd">
```

A palavra-chave `SYSTEM` indica que o XML parser deve localizar o DTD de uma maneira específica no sistema. Isso normalmente (mas nem sempre) significa que o DTD será fornecido como um nome de arquivo.

Usando FPIs é preferível por razões de portabilidade. Se o identificador `SYSTEM` for usado, então o DTD deve ser fornecido e mantido no mesmo local para todos.

7.4. De Volta para o XML

Algumas das sintaxes subjacentes do XML podem ser úteis em documentos. Por exemplo, os comentários podem ser incluídos no documento e serão ignorados pelo parser. Os comentários são inseridos usando a sintaxe XML. Outros usos para a sintaxe XML serão mostrados mais tarde.

Seções XML começam com uma tag `<!` e terminam com `>`. Essas seções contêm instruções para o parser em vez de elementos do documento. Tudo entre essas tags são sintaxe XML. A declaração DOCTYPE mostrada anteriormente é um exemplo da sintaxe XML incluída no documento.

7.5. Comentários

Um documento XML pode conter comentários. Eles podem aparecer em qualquer lugar, desde que não estejam dentro de tags. Eles até são permitidos em alguns locais dentro do DTD (por exemplo, entre declarações de entidade).

Os comentários XML começam com a string "`<!--`" e terminam com a string "`-->`".

Aqui estão alguns exemplos de comentários válidos de XML:

Exemplo 7.6. XML Comentários Genéricos

```
<!-- This is inside the comment -->

<!--This is another comment-->

<!-- This is how you
     write multiline comments -->

<p>A simple <!-- Comment inside an element's content --> paragraph.</p>
```

Os comentários XML podem conter quaisquer strings, exceto "`--`":

Exemplo 7.7. Comentário XML Incorreto

```
<!-- This comment--is wrong -->
```

7.5.1. Para Fazer...

1. Adicione alguns comentários ao arquivo `example.xml` e depois o valide utilizando o `xmllint`.
2. Adicione alguns comentários inválidos ao arquivo `example.xml` e veja as mensagens de erros que o `xmllint` irá retornar quando encontrar algum comentário inválido.

7.6. Entidades

Entidades são um mecanismo para atribuir nomes a partes do conteúdo. À medida que um XML parser processa um documento, qualquer entidade encontrada é substituída pelo conteúdo da entidade.

Esta é uma boa maneira de ter pedaços de conteúdo reutilizáveis e facilmente alteráveis em documentos XML. Também é a única maneira de incluir um arquivo markup dentro de outro usando XML.

Existem dois tipos de entidades para duas situações diferentes: *entidades gerais* e *entidades de parâmetros*.

7.6.1. Entidades Gerais

Entidades gerais são usadas para atribuir nomes a partes reutilizáveis de texto. Essas entidades só podem ser usadas no documento. Elas não podem ser usadas em um contexto XML.

Para incluir o texto de uma entidade geral no documento, inclua `&`*`nome-da-entidade`*` ;` no texto. Por exemplo, considere uma entidade geral chamada `current.version`, que se expande para o número da versão atual de um produto. Para usá-la no documento, escreva:

```
<para>The current version of our product is
  &current.version;.</para>
```

Quando o número da versão for alterado, edite a definição da entidade geral, substituindo o valor. Em seguida, reprocesse o documento.

Entidades gerais também podem ser usadas para inserir caracteres que não poderiam ser incluídos em um documento XML. Por exemplo, < e & normalmente não podem aparecer em um documento XML. O XML parser vê o símbolo < como o início de uma tag. Da mesma forma, quando o símbolo & é visto, espera-se que o próximo texto seja um nome de entidade.

Esses símbolos podem ser incluídos usando duas entidades gerais predefinidas: `<` e `&`.

Entidades gerais só podem ser definidas dentro de um contexto XML. Tais definições geralmente são feitas imediatamente após a declaração DOCTYPE.

Exemplo 7.8. Definindo Entidades Gerais

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" [
<!ENTITY current.version    "3.0-RELEASE">
<!ENTITY last.version       "2.2.7-RELEASE">
]>
```

A declaração DOCTYPE foi estendida adicionando um colchete no final da primeira linha. As duas entidades são então definidas nas próximas duas linhas, o colchete é fechado e, em seguida, a declaração DOCTYPE é fechada.

Os colchetes são necessários para indicar que o DTD indicado pela declaração DOCTYPE está sendo estendido.

7.6.2. Entidades de Parâmetro

Entidades de parâmetro, como as entidades gerais, são usadas para atribuir nomes a blocos reutilizáveis de texto. Mas as entidades de parâmetro só podem ser usadas dentro de um contexto XML.

As definições de entidade de parâmetro são semelhantes àquelas para entidades gerais. No entanto, entradas de parâmetros são incluídas com `%nome-da-entidade ;`. A definição também inclui o `%` entre a palavra-chave `ENTITY` e o nome da entidade.

Para memorizar, lembre que entidade de"Parâmetro utiliza o símbolo de Porcentagem".

Exemplo 7.9. Definindo Entidades de Parâmetro

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" [
<!ENTITY % entity "<!ENTITY version '1.0'>">
<!-- use the parameter entity -->
%entity;
]>
```

À primeira vista, as entidades de parâmetros não parecem muito úteis, mas elas tornam possível incluir outros arquivos em um documento XML.

7.6.3. Para Fazer...

1. Adicione uma entidade geral ao arquivo `example.xml` .

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" [
<!ENTITY version "1.1">
]>

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>An Example XHTML File</title>
  </head>

  <!-- There may be some comments in here as well -->

  <body>
    <p>This is a paragraph containing some text.</p>

    <p>This paragraph contains some more text.</p>

    <p align="right"> This paragraph might be right-justified.</p>

    <p>The current version of this document is: &version;</p>
  </body>
</html>
```

2. Valide o documento usando o `xmllint` .

3. Carregue `example.xml` em um navegador web. Ele pode ter que ser copiado para o `example.html` antes que o navegador o reconheça como um documento XHTML.

   Navegadores mais antigos com parsers simples podem não renderizar esse arquivo conforme o esperado. A referência de entidade `&version;` pode não ser substituída pelo número da versão, ou o fechamento de contexto XML `]>` pode não ser reconhecido e em vez disso, apresentado literalmente.

4. A solução é *normalizar* o documento com um normalizador XML. O normalizador lê um XML válido e grava outro XML igualmente válido. Uma maneira pela qual o normalizador transforma a entrada é expandindo todas as referências de entidade no documento, substituindo as entidades pelo texto que elas representam.

   O `xmllint` pode ser usado para isso. Ele também tem a opção de remover a seção inicial DTD para que `]>` não confunda os navegadores:

```
% xmllint --noent --dropdtd example.xml > example.html
```

   Uma cópia normalizada do documento com entidades expandidas é produzida em `example.html` , pronta para ser carregada em um navegador web.

7.7. Usando Entidades para Incluir Arquivos

Ambas as entidades geral e parâmetro são particularmente úteis para incluir um arquivo dentro de outro .

7.7.1. Usando Entidades Gerais para Incluir Arquivos

Considere algum conteúdo para um livro XML organizado em arquivos, um arquivo por capítulo, chamado `chapter1.xml` , `chapter2.xml` e assim por diante, com um `book.xml` que conterá esses capítulos.

Para usar o conteúdo desses arquivos como valores para entidades, eles são declarados com a palavra-chave `SYSTEM`. Isso direciona o XML parser a incluir o conteúdo do arquivo nomeado como o valor da entidade.

Exemplo 7.10. Usando Entidades Gerais para Incluir Arquivos

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" [
<!ENTITY chapter.1 SYSTEM "chapter1.xml">
<!ENTITY chapter.2 SYSTEM "chapter2.xml">
<!ENTITY chapter.3 SYSTEM "chapter3.xml">
<!-- And so forth -->
]>

<html xmlns="http://www.w3.org/1999/xhtml">
  <!-- Use the entities to load in the chapters -->

  &chapter.1;
  &chapter.2;
  &chapter.3;
</html>
```

Atenção

Ao usar entidades gerais para incluir outros arquivos em um documento, os arquivos que estão sendo incluídos (`chapter1.xml` , `chapter2.xml` e assim por diante) *não devem* começar com uma declaração DOCTYPE. Este é um erro de sintaxe porque as entidades são constructors de baixo nível e são transformadas antes que qualquer análise ocorra.

7.7.2. Usando Entidades de Parâmetro para Incluir Arquivos

As entidades de parâmetro só podem ser usadas dentro de um contexto XML. A inclusão de um arquivo em um contexto XML pode ser usada para garantir que as entidades gerais sejam reutilizáveis.

Suponha que haja muitos capítulos no documento, e esses capítulos foram reutilizados em dois livros diferentes, cada livro organizando os capítulos de maneira diferente.

As entidades podem ser listadas no topo de cada livro, mas isso rapidamente se torna difícil de gerenciar.

Em vez disso, coloque as definições gerais da entidade em um arquivo e use uma entidade de parâmetro para incluir esse arquivo no documento.

Exemplo 7.11. Usando Entidades de Parâmetro para Incluir Arquivos

Coloque as definições de entidade em um arquivo separado chamado `chapters.ent` contendo este texto:

```
<!ENTITY chapter.1 SYSTEM "chapter1.xml">
<!ENTITY chapter.2 SYSTEM "chapter2.xml">
<!ENTITY chapter.3 SYSTEM "chapter3.xml">
```

Crie uma entidade de parâmetro para se referir ao conteúdo do arquivo. Em seguida, use a entidade de parâmetro para carregar o arquivo no documento, o que tornará todas as entidades gerais disponíveis para uso. Em seguida, use as entidades gerais como antes:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" [
<!-- Define a parameter entity to load in the chapter general entities -->
```

```
<!ENTITY % chapters SYSTEM "chapters.ent">

<!-- Now use the parameter entity to load in this file -->
%chapters;
]>

<html xmlns="http://www.w3.org/1999/xhtml">
  &chapter.1;
  &chapter.2;
  &chapter.3;
</html>
```

7.7.3. Para Fazer...

7.7.3.1. Use Entidades Gerais para Incluir Arquivos

1. Crie três arquivos, `para1.xml` , `para2.xml` e `para3.xml` .

   Coloque conteúdo como este em cada arquivo:

   ```
   <p>This is the first paragraph.</p>
   ```

2. Edite `example.xml` para que fique assim:

   ```
   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" [
   <!ENTITY version "1.1">
   <!ENTITY para1 SYSTEM "para1.xml">
   <!ENTITY para2 SYSTEM "para2.xml">
   <!ENTITY para3 SYSTEM "para3.xml">
   ]>

   <html xmlns="http://www.w3.org/1999/xhtml">
     <head>
       <title>An Example XHTML File</title>
     </head>

     <body>
       <p>The current version of this document is: &version;</p>

       &para1;
       &para2;
       &para3;
     </body>
   </html>
   ```

3. Gere `example.html` ao normalizar `example.xml` .

   ```
   % xmllint --dropdtd --noent example.xml > example.html
   ```

4. Carregue `example.html` no navegador web e confirme se os arquivos `para`*n*`.xml` foram incluídos em `example.html` .

7.7.3.2. Use Entidades de Parâmetro para Incluir Arquivos

Nota

As etapas anteriores devem ser concluídas antes dessa etapa.

1. Edite `example.xml` para que fique assim:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" [
<!ENTITY % entities SYSTEM "entities.ent"> %entities;
]>

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>An Example XHTML File</title>
  </head>

  <body>
    <p>The current version of this document is: &version;</p>

    &para1;
    &para2;
    &para3;
  </body>
</html>
```

2. Crie um novo arquivo chamado `entities.ent` com este conteúdo:

```
<!ENTITY version "1.1">
<!ENTITY para1 SYSTEM "para1.xml">
<!ENTITY para2 SYSTEM "para2.xml">
<!ENTITY para3 SYSTEM "para3.xml">
```

3. Gere `example.html` ao normalizar `example.xml`.

```
% xmllint --dropdtd --noent example.xml > example.html
```

4. Carregue `example.html` no navegador web e confirme se os arquivos `para`*n*`.xml` foram incluídos em `example.html`.

7.8. Seções Marcadas

XML fornece um mecanismo para indicar que partes específicas do documento devem ser processadas de uma maneira especial. Estes são chamados de “seções marcadas”.

Exemplo 7.12. Estrutura de uma Seção Marcada

```
<![KEYWORD [
  Contents of marked section
]]>
```

Como esperado de um construct XML, uma seção marcada começa com `<!`.

O primeiro colchete inicia a seção marcada.

`KEYWORD` descreve como esta seção marcada deve ser processada pelo parser.

O segundo colchete indica o início do conteúdo da seção marcada.

A seção marcada é concluída, fechando os dois colchetes e, em seguida, retornando ao contexto do documento do contexto XML com `>`.

7.8.1. Palavras-chave da Seção Marcada

7.8.1.1. CDATA

Essas palavras-chave indicam as seções marcadas pelo *modelo de conteúdo* e permitem que você a altere do padrão.

Quando um XML parser está processando um documento, ele acompanha o "modelo de conteúdo".

O modelo de conteúdo descreve o conteúdo que o parser espera ver e o que ele fará com esse conteúdo.

O modelo de conteúdo `CDATA` é um dos mais úteis.

`CDATA` é para "Dados de Caractere". Quando o parser está neste modelo de conteúdo, ele espera ver apenas caracteres. Nesse modelo, os símbolos < e & perdem seu status especial e serão tratados como caracteres comuns.

Nota

Ao usar `CDATA` em exemplos de texto marcados em XML, lembre-se de que o conteúdo de `CDATA` não é validado. O texto incluído deve ser verificado por outros meios. Por exemplo, o conteúdo poderia ser escrito em outro documento, validado e depois colado na seção `CDATA`.

Exemplo 7.13. Usando uma Seção Marcada `CDATA`

```
<para>Here is an example of how to include some text that contains
  many <literal>&lt;</literal>  and <literal>&amp;</literal>
  symbols.  The sample text is a fragment of
  <acronym>XHTML</acronym>.  The surrounding text (<para> and
  <programlisting> ) are from DocBook.</para>

<programlisting> <![CDATA[<p>This is a sample that shows some of the
  elements within <acronym>XHTML</acronym>.  Since the angle
  brackets are used so many times, it is simpler to say the whole
  example is a CDATA marked section than to use the entity names for
  the left and right angle brackets throughout.</p>

    <ul>
      <li>This is a listitem</li>
      <li>This is a second listitem</li>
      <li>This is a third listitem</li>
    </ul>

    <p>This is the end of the example.</p>]]></programlisting>
```

7.8.1.2. INCLUDE e IGNORE

Quando a palavra-chave é `INCLUDE`, o conteúdo da seção marcada será processado. Quando a palavra-chave é `IGNORE`, a seção marcada é ignorada e não será processada. Não aparecerá na saída.

Exemplo 7.14. Usando `INCLUDE` e `IGNORE` em Seções Marcadas

```
<![INCLUDE[
  This text will be processed and included.
]]>
```

```
<![IGNORE[
  This text will not be processed or included.
]]>
```

Por si só, isso não é muito útil. O texto a ser removido do documento pode ser recortado ou estar em forma de comentários.

Ele se torna mais útil quando controlado por entidades de parâmetro, mas esse uso é limitado a arquivos de entidades.

Por exemplo, suponha que a documentação tenha sido produzida em uma versão impressa e em uma versão eletrônica. Algum texto extra é desejado no conteúdo da versão eletrônica que não deveria aparecer na cópia impressa.

Crie um arquivo de entidade que defina entidades gerais para incluir cada capítulo e proteja essas definições com uma entidade de parâmetro que pode ser definida como `INCLUDE` ou `IGNORE` para controlar se a entidade está definida . Após essas definições de entidades gerais condicionais, coloque mais uma definição para cada entidade geral para defini-las como um valor vazio. Essa técnica faz uso do fato de que as definições de entidade não podem ser substituídas, mas a primeira definição sempre entra em vigor. Assim, a inclusão do capítulo é controlada com a entidade de parâmetro correspondente. Definido como `INCLUDE` , a primeira definição de entidade geral será lida e a segunda será ignorada. Definido como `IGNORE` , a primeira definição será ignorada e a segunda será utilizada.

Exemplo 7.15. Usando uma Entidade de Parâmetro para Controlar uma Seção Marcada

```
<!ENTITY % electronic.copy "INCLUDE">

<![%electronic.copy;[
<!ENTITY chap.preface SYSTEM "preface.xml">
]]>

<!ENTITY chap.preface "">
```

Ao produzir a versão impressa, altere a definição do parâmetro de entidade para:

```
<!ENTITY % electronic.copy "IGNORE">
```

7.8.2. Para Fazer...

1. Modifique `entidades.ent` para conter o seguinte texto:

   ```
   <!ENTITY version "1.1">
   <!ENTITY % conditional.text "IGNORE">

   <![%conditional.text;[
   <!ENTITY para1 SYSTEM "para1.xml">
   ]]>

   <!ENTITY para1 "">

   <!ENTITY para2 SYSTEM "para2.xml">
   <!ENTITY para3 SYSTEM "para3.xml">
   ```

2. Normalize `example.xml` e observe que o texto condicional não está presente no documento de saída. Altere o parâmetro de entidade para `INCLUDE` e gere novamente o documento normalizado, dessa forma e o texto

aparecerá novamente. Esse método faz sentido se houver mais partes condicionais dependendo da mesma condição. Por exemplo, para controlar a geração de texto impresso ou on-line.

7.9. Conclusão

Essa é a conclusão deste primer XML. Por razões de espaço e complexidade, vários assuntos não foram cobertos bem afundo. No entanto, as seções anteriores abrangem o suficiente de XML para apresentar a organização da documentação do FDP.

Capítulo 8. XHTML Markup

8.1. Introdução

Este capítulo descreve o uso da linguagem XHTML markup usada no site do FreeBSD.

XHTML é a versão XML da HyperText Markup Language, a linguagem markup escolhida na World Wide Web. Mais informações podem ser encontradas em `http://www.w3.org/` .

XHTML é usado para escrever páginas no site do FreeBSD. Geralmente não é usado para escrever outra documentação, uma vez que o DocBook oferece um conjunto muito mais rico de elementos para se escolher. Consequentemente, as páginas XHTML normalmente só serão encontradas ao escrever para o web site.

O HTML passou por várias versões. A versão compatível com XML descrita aqui é chamada XHTML. A versão mais recente generalizada é o XHTML 1.0, disponível nas variantes *strict* e *transitional*.

Os XHTML DTDs estão disponíveis na Coleção de Ports em textproc/xhtml. Eles são automaticamente instalados pelo port textproc/docproj.

Nota

Isto *não* é uma lista completa de elementos, uma vez que isso apenas repetiria a documentação de XHTML. O objetivo é listar os elementos mais utilizados. Por favor, poste perguntas sobre elementos ou usos não abordados aqui na lista de discussão do projeto de documentação do FreeBSD.

Inline Versus Block

No restante deste documento, ao descrever elementos, *inline* significa que o elemento pode estar dentro de um elemento de bloco e não causa uma quebra de linha. Um elemento *block*, por outro lado, causará uma quebra de linha (e outro processamento) quando for encontrado.

8.2. Identificador Público Formal (FPI)

Existem vários XHTML FPIs, dependendo da versão, ou da *versão* do XHTML ao qual um documento está em conformidade. A maioria dos documentos XHTML no site do FreeBSD está de acordo com a versão de transição do XHTML 1.0.

```
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```

8.3. Seções de Elementos

Um documento XHTML é normalmente dividido em duas seções. A primeira seção, chamada de *head*, contém meta-informações sobre o documento, como seu título, o nome do autor, o documento pai e assim por diante. A segunda seção, o*body*, contém o conteúdo que será exibido ao usuário.

Essas seções são indicadas com os elementos `head` e `body`, respectivamente. Esses elementos estão dentro do elemento `html` de nível superior.

Exemplo 8.1. Estrutura de um Documento XHTML

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
   <title>The Document's Title </title>
  </head>

  <body>

    …

  </body>
</html>
```

8.4. Elementos Block

8.4.1. Cabeçalhos

XHTML tem tags para indicar títulos no documento em até seis níveis diferentes.

O maior e mais importante título é `h1`, depois `h2`, seguindo até `h6`.

O conteúdo do elemento é o texto do cabeçalho.

Exemplo 8.2. `h1`, `h2` e Outras Tags de Cabeçalho

Uso:

```
<h1>First section</h1>

<!-- Document introduction goes here -->

<h2>This is the heading for the first section</h2>

<!-- Content for the first section goes here -->

<h3>This is the heading for the first sub-section</h3>

<!-- Content for the first sub-section goes here -->

<h2>This is the heading for the second section</h2>

<!-- Content for the second section goes here -->
```

Geralmente, uma página XHTML deve ter um título de primeiro nível (`h1`). Nela pode conter muitos títulos de segundo nível (`h2`), que por sua vez podem conter muitos cabeçalhos de terceiro nível. Não deixe lacunas na numeração.

8.4.2. Parágrafos

O XHTML suporta um único elemento de parágrafo, `p`.

Exemplo 8.3. Exemplo `p`

Uso:

```
<p>This is a paragraph.  It can contain just about any
  other element.</p>
```

8.4.3. Bloco de Citações

Um bloco de citação é uma citação estendida de outro documento que aparecerá em um parágrafo separado.

Exemplo 8.4. Exemplo `blockquote`

Uso:

```
<p>A small excerpt from the US Constitution:</p>

<blockquote> We the People of the United States, in Order to form
  a more perfect Union, establish Justice, insure domestic
  Tranquility, provide for the common defence, promote the general
  Welfare, and secure the Blessings of Liberty to ourselves and our
  Posterity, do ordain and establish this Constitution for the
  United States of America.</blockquote>
```

8.4.4. Listas

O XHTML pode apresentar ao usuário três tipos de listas: ordenadas, desordenadas e de definição.

Entradas em uma lista ordenada serão numeradas, enquanto as entradas em uma lista não ordenada serão precedidas por marcadores. Listas de definições têm duas seções para cada entrada. A primeira seção é o termo que está sendo definido e a segunda seção é a definição.

As listas ordenadas são indicadas pelo elemento `ol`, listas não ordenadas pelo elemento `ul` e listas de definição pelo elemento `dl`.

As listas ordenadas e não ordenadas contêm listitens, indicadas pelo elemento `li`. Um listitem pode conter conteúdo textual ou pode ser envoltos em um ou mais elementos `p`.

As listas de definições contêm termos de definição (`dt`) e descrições de definição (`dd`). Um termo de definição pode conter apenas elementos in-line. Uma descrição de definição pode conter outros elementos de bloco.

Exemplo 8.5. Exemplo `ul` and `ol`

Uso:

```
<p>An unordered list.  Listitems will probably be
  preceded by bullets.</p>

<ul>
  <li>First item</li>
```

```
  <li>Second item</li>

  <li>Third item</li>
</ul>

<p>An ordered list, with list items consisting of multiple
  paragraphs.  Each item (note: not each paragraph) will be
  numbered.</p>

<ol>
  <li><p>This is the first item.  It only has one paragraph.</p></li>

  <li><p>This is the first paragraph of the second item.</p>

    <p>This is the second paragraph of the second item.</p></li>

  <li><p>This is the first and only paragraph of the third
    item.</p></li>
</ol>
```

Exemplo 8.6. Listas de Definição com `dl`

Uso:

```
<dl>
  <dt>Term 1</dt>

  <dd><p>Paragraph 1 of definition 1.</p>

    <p>Paragraph 2 of definition 1.</p></dd>

  <dt>Term 2</dt>

  <dd><p>Paragraph 1 of definition 2.</p></dd>

  <dt>Term 3</dt>

  <dd><p>Paragraph 1 of definition 3.</p></dd>
</dl>
```

8.4.5. Texto Pré-formatado

Texto pré-formatado é apresentado para o usuário exatamente como está no arquivo. O texto é mostrado em uma fonte fixa. Vários espaços e quebras de linha são mostrados exatamente como estão no arquivo.

Deixe o texto pré-formatado no elemento `pre`.

Exemplo 8.7. Exemplo `pre`

Por exemplo, as tags `pre` podem ser usadas para marcar uma mensagem de email:

```
<pre>  From: nik@FreeBSD.org
  To: freebsd-doc@FreeBSD.org
  Subject: New documentation available
```

```
  There is a new copy of my primer for contributors to the FreeBSD
  Documentation Project available at

    &lt;URL:https://people.FreeBSD.org/~nik/primer/index.html&gt;

  Comments appreciated.

  N</pre>
```

Tenha em mente que < e & ainda são reconhecidos como caracteres especiais no texto pré-formatado. É por isso que o exemplo mostrado teve que usar `<` em vez de <. Para consistência, o `>` também foi usado no lugar de >. Fique atento com os caracteres especiais que podem aparecer no texto copiado de uma fonte de texto simples, como uma mensagem de e-mail ou código de programa.

8.4.6. Tabelas

Marque as informações tabulares usando o elemento `table`. Uma tabela consiste em uma ou mais linhas da tabela (`tr`), cada uma contendo uma ou mais células de dados da tabela (`td`). Cada célula pode conter outros elementos de bloco, como parágrafos ou listas. Também pode conter outra tabela (esse aninhamento pode repetir indefinidamente). Se a célula contiver apenas um parágrafo, o elemento `p` não será necessário.

Exemplo 8.8. Uso Simples de `table`

Uso:

```
<p>This is a simple 2x2 table.</p>

<table>
  <tr>
    <td>Top left cell</td>

    <td>Top right cell</td>
  </tr>

  <tr>
    <td>Bottom left cell</td>

    <td>Bottom right cell</td>
  </tr>
</table>
```

Uma célula pode abranger várias linhas e colunas adicionando os atributos `rowspan` ou `colspan` com valores para o número de linhas ou colunas a serem abrangido.

Exemplo 8.9. Usando `rowspan`

Uso:

```
<p>One tall thin cell on the left, two short cells next to
  it on the right.</p>

<table>
  <tr>
```

```
    <td rowspan="2"> Long and thin</td>
  </tr>

  <tr>
    <td>Top cell</td>

    <td>Bottom cell</td>
  </tr>
</table>
```

Exemplo 8.10. Usando `colspan`

Uso:

```
<p>One long cell on top, two short cells below it.</p>

<table>
  <tr>
    <td colspan="2"> Top cell</td>
  </tr>

  <tr>
    <td>Bottom left cell</td>

    <td>Bottom right cell</td>
  </tr>
</table>
```

Exemplo 8.11. Usando `rowspan` e `colspan` Juntos

Uso:

```
<p>On a 3x3 grid, the top left block is a 2x2 set of
  cells merged into one.  The other cells are normal.</p>

<table>
  <tr>
    <td colspan="2" rowspan="2"> Top left large cell</td>

    <td>Top right cell</td>
  </tr>

  <tr>
    <!-- Because the large cell on the left merges into
         this row, the first <td> will occur on its
         right -->

    <td>Middle right cell</td>
  </tr>

  <tr>
    <td>Bottom left cell</td>

    <td>Bottom middle cell</td>

    <td>Bottom right cell</td>
```

```
  </tr>
</table>
```

8.5. Elementos In-line

8.5.1. Realçando Informação

Dois níveis de ênfase estão disponíveis em XHTML, `em` e `strong`. `em` é para um nível normal de ênfase e `strong` indica ênfase mais forte.

`em` é normalmente renderizado em itálico e o `strong` é renderizado em negrito. Isso nem sempre assim e não deve ser considerado ao pé da letra. De acordo com as práticas recomendadas, as páginas web armazenam apenas informações estruturais e semânticas, e as folhas de estilo são aplicadas posteriormente a elas. Pense na semântica, não na formatação, ao usar essas tags.

Exemplo 8.12. Exemplo `em` e `strong`

Uso:

```
<p><em>This</em> has been emphasized, while
  <strong> this</strong> has been strongly emphasized.</p>
```

8.5.2. Indicando Texto Fixo

O conteúdo que deve ser renderizado em um tipo fixo de texto (typewriter) é marcado com `tt` (para “teletype”).

Exemplo 8.13. Exemplo `tt`

Uso:

```
<p>Many system settings are stored in
  <tt>/etc</tt>.</p>
```

8.5.3. Links

Nota

Links também são elementos in-line.

8.5.3.1. Criando Links para Outros Documentos na Web

Um link aponta para uma URL de um documento na web. O link é indicado com a `a`, e o atributo `href` contém a URL do documento de destino. O conteúdo do elemento se torna o link, indicado ao usuário, mostrando-o em uma cor diferente ou com um sublinhado.

Exemplo 8.14. Usando `<a href="...">`

Uso:

```
<p>More information is available at the
  <a href="http://www.&os;.org/">  &os; web site</a>.</p>
```

Esse link sempre leva o usuário ao topo do documento que foi linkado.

8.5.3.2. Criando Links para Partes Específicas de Documentos

Para linkar a um ponto específico dentro de um documento, esse documento deve incluir uma *âncora* no ponto desejado. As âncoras são incluídas configurando o atributo `id` de um elemento para um nome. Este exemplo cria uma âncora definindo o atributo `id` de um elemento `p`.

Exemplo 8.15. Criando uma Âncora

Uso:

```
<p id="samplepara"> This paragraph can be referenced
  in other links with the name <tt>samplepara</tt>.</p>
```

Links para âncoras são semelhantes aos links simples, mas incluem um símbolo # e o ID da âncora no final da URL.

Exemplo 8.16. Criando Link para uma Parte Nomeada de um Outro Documento

O exemplo `samplepara` é parte de um documento chamado `foo.html`. Um link para esse parágrafo específico no documento é construído neste exemplo.

```
<p>More information can be found in the
  <a href="foo.html#samplepara">  sample paragraph</a> of
  <tt>foo.html</tt>.</p>
```

Para vincular-se a uma âncora nomeada no mesmo documento, omita a URLdo documento, e use apenas o símbolo # seguido do nome da âncora.

Exemplo 8.17. Criando Link para uma Parte Nomeada no Mesmo Documento

O exemplo `samplepara` reside neste documento. Para vincular a ele:

```
<p>More information can be found in the
  <a href="#samplepara"> sample paragraph</a> of this
```

```
  document.</p>
```

Capítulo 9. DocBook Markup

9.1. Introdução

Este capítulo é uma introdução ao DocBook e como ele é usado na documentação do FreeBSD. O DocBook é um sistema de marcação extenso e complexo, mas o subconjunto descrito aqui abrange as partes mais usadas para a documentação do FreeBSD. Enquanto um subconjunto moderado é coberto, é impossível antecipar todas as situações. Por favor, poste questões que este documento não responde à lista de discussão do projeto de documentação do FreeBSD.

DocBook foi originalmente desenvolvido por HaL Computer Systems and O'Reilly & Associates para ser uma Definição de Tipo de Documento (DTD) para escrever documentação técnica [1]. Desde 1998, ele é mantido pelo Comitê Técnico do DocBook. Como tal, e ao contrário do LinuxDoc e do XHTML, o DocBook é fortemente orientado para um markup que descreve *o que* alguma coisa é, em vez de descrever *como* deve ser apresentado.

O DocBook DTD está disponível na coleção Ports textproc/docbook-xml. Ele é instalado automaticamente como parte do port textproc/docproj.

Formal Versus Informal

Alguns elementos podem existir em duas formas, *formal* e *informal*. Normalmente, a versão formal do elemento consistirá em um título seguido pela versão informal do elemento. A versão informal não terá um título.

Inline Versus Block

No restante deste documento, ao descrever elementos, *inline* significa que o elemento pode estar dentro de um elemento de bloco e não causa uma quebra de linha. Um elemento *block*, por outro lado, causará uma quebra de linha (e outro processamento) quando for encontrado.

9.2. Extensões do FreeBSD

O Projeto de Documentação do FreeBSD estendeu o DocBook DTD com elementos e entidades adicionais. Essas adições servem para tornar algumas das marcações mais fáceis ou mais precisas.

Ao longo deste documento, o termo “DocBook” é usado para indicar o DocBook DTD estendido do FreeBSD.

Nota

A maioria dessas extensões não é exclusiva do FreeBSD, foi apenas identificado que elas eram melhorias úteis para este projeto. Se alguém de qualquer um dos outros *nix (NetBSD, OpenBSD, Linux,...) estiverem interessados em colaborar em um conjunto de

[1] Um breve histórico pode ser encontrado em http://www.oasis-open.org/docbook/intro.shtml#d0e41.

extensão DocBook padrão, entre em contato com a Equipe de Engenharia de Documentação <doceng@FreeBSD.org >.

9.2.1. Elementos do FreeBSD

Os elementos adicionais do FreeBSD não estão (atualmente) na Coleção de Ports. Eles são armazenados na repositório Subversion do FreeBSD, como head/share/xml/freebsd.dtd.

Os elementos específicos do FreeBSD usados nos exemplos abaixo estão claramente marcados.

9.2.2. Entidades do FreeBSD

Esta tabela mostra algumas das entidades mais úteis disponíveis no FDP. Para obter uma lista completa, consulte os arquivos *.ent em doc/share/xml .

Entidades de Nome do FreeBSD		
&os;	FreeBSD	
&os.stable;	FreeBSD-STABLE	
&os.current;	FreeBSD-CURRENT	
Entidades de Página de Manual		
&man.ls.1;	ls(1)	Uso: &man.ls.1; é a página de manual para o <command>ls</command>.
&man.cp.1;	cp(1)	Uso: A página de manual para <command>cp</command> é &man.cp.1;.
&man. *command* .*sectionnumber* ;	*link para a página de manual* *command* *na seção* *sectionnumber*	As entidades são definidas para todas as páginas de manual do FreeBSD.
Entidades das Listas de Email do FreeBSD		
&a.doc;	Lista de discussão do projeto de documentação do FreeBSD	Uso: Um link para a &a.doc;.
&a.questions;	Lista de discussão de assuntos gerais do FreeBSD	Uso: Um link para a &a.questions;.
&a. *listname* ;	*link para* *listname*	Foram criadas entidades para todas as listas de email do FreeBSD.
Entidades de Link de Documento do FreeBSD		
&url.books.handbook; a	https://www.FreeBSD.org/doc/en_US.ISO8859-1/books/handbook	Uso: Um link para o capítulo <link xlink:href="&url.books.handbook;/advanced-networking.html">Rede Avançado</link> do Handbook.
&url.books. *bookname* ;	*caminho relativo para* *bookname*	As entidades são definidas para todos os livros FreeBSD.

&url.articles.committers-guide;	https://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/committers-guide	Uso: Um link para o artigo <link xlink:href="&url.articles.committers-guide;">Guia dos Committer's</link>.
&url.articles. *articlename* ;	*caminho relativo para* articlename	As entidades são definidas para todos os artigos do FreeBSD.
Outras Entidades de Sistema Operacional		
&linux;	Linux®	O sistema operacional Linux®.
&unix;	UNIX®	O sistema operacional UNIX®.
&windows;	Windows®	O sistema operacional Windows®.
Entidades Diversas		
&prompt.root;	#	O prompt de usuário root .
&prompt.user;	%	Um prompt para um usuário sem privilégios.
&postscript;	PostScript®	A linguagem de programação PostScript®.
&tex;	TeX	A linguagem de composição tipográfica TeX.
&xorg;	Xorg	Xorg, Sistema X Window de código aberto.

9.3. Identificador Público Formal (FPI)

Em conformidade com as diretrizes do DocBook, para escrever FPIs customizadas do DocBook, o FPI para o DocBook DTD estendido do FreeBSD é:

```
PUBLIC "-//FreeBSD//DTD DocBook V4.2-Based Extension//EN"
```

9.4. Estrutura do Documento

O DocBook permite a documentação de estruturação de várias maneiras. O Projeto de Documentação do FreeBSD usa dois tipos principais de documentos do DocBook: o book e o article.

Os livros (books) são organizados em chapter s. Este é um requisito obrigatório. Pode haver partes entre o livro e o capítulo para fornecer outra camada de organização. Por exemplo, o Handbook é organizado dessa maneira.

Um capítulo(chapter) pode (ou não) conter uma ou mais seções. Estes são indicados com o elemento sect1. Se uma seção contiver outra seção, use o elemento sect2 e assim por diante, até sect5.

Capítulos e seções contêm o restante do conteúdo.

Um artigo é mais simples que um livro e não utiliza capítulos. Em vez disso, o conteúdo de um artigo é organizado em uma ou mais seções, usando os mesmos elementos sect1 (e sect2 e assim por diante) que são usados em livros.

A natureza do documento que está sendo escrito deve ser usada para determinar se ele é melhor marcado como um livro ou um artigo. Os artigos são adequados à informação que não precisa ser dividida em vários capítulos, ou

seja, relativamente curta, em até 20-25 páginas de conteúdo. Os livros são mais adequados para informações que podem ser divididas em vários capítulos, possivelmente com apêndices e conteúdo similar também.

Os Tutoriais do FreeBSD estão todos marcados como artigos, enquanto este documento, o FAQ e o Handbook estão todos marcados como livros, por exemplo.

9.4.1. Iniciando um Livro

O conteúdo de um livro está dentro do elemento `book`. Além de conter marcação estrutural, esse elemento pode conter elementos que incluem informações adicionais sobre o livro. Isso é uma meta-informação, usada para fins de referência ou conteúdo adicional usado para produzir um título de página.

Esta informação adicional está em um elemento `info`.

Exemplo 9.1. Exemplo de `book` com `info`

```
<book>
  <info>
    <title>Your Title Here</title>

    <author>
      <personname>
        <firstname>Your first name</firstname>
        <surname>Your surname</surname>
      </personname>

      <affiliation>
 <address>
          <email>Your email address</email>
 </address>
      </affiliation>
    </author>

    <copyright>
      <year>1998</year>
      <holder role="mailto: your email address ">Your name</holder>
    </copyright>

    <releaseinfo> $FreeBSD$</releaseinfo>

    <abstract>
      <para>Include an abstract of the book's contents here.  </para>
    </abstract>
  </info>

  …

</book>
```

9.4.2. Iniciando um Artigo

O conteúdo do artigo está dentro do elemento `article`. Além de conter marcação estrutural, esse elemento pode conter elementos que incluem informações adicionais sobre o artigo. Isso é uma meta-informação, usada para fins de referência ou conteúdo adicional usado para produzir um título de página.

Esta informação adicional está em um elemento `info`.

Exemplo 9.2. Exemplo de `article` com `info`

```
<article>
  <info>
    <title>Your title here</title>

    <author>
      <personname>
 <firstname>Your first name</firstname>
 <surname>Your surname</surname>
      </personname>

      <affiliation>
 <address>
   <email>Your email address</email></address>
 </address>
      </affiliation>
    </author>

    <copyright>
      <year>1998</year>
      <holder role="mailto: your email address ">Your name</holder>
    </copyright>

    <releaseinfo> $FreeBSD$</releaseinfo>

    <abstract>
      <para>Include an abstract of the article's contents here.  </para>
    </abstract>
  </info>

  …

</article>
```

9.4.3. Criando Capítulos

Use `chapter` para marcar seus capítulos. Cada capítulo tem um `title` obrigatório. Os artigos não contêm capítulos, eles são reservados para livros.

Exemplo 9.3. Um Capítulo Simples

```
<chapter>
  <title>The Chapter's Title</title>

  ...
</chapter>
```

Um capítulo não pode estar vazio; ele deve conter elementos além do `title`. Se você precisar incluir um capítulo vazio, basta usar um parágrafo vazio.

Exemplo 9.4. Capítulos Vazios

```
<chapter>
  <title>This is An Empty Chapter</title>

  <para></para>
</chapter>
```

9.4.4. Seções Abaixo dos Capítulos

Nos livros, os capítulos podem (mas não precisam) ser divididos em seções, subseções e assim por diante. Nos artigos, as seções são o principal elemento estrutural e cada artigo deve conter pelo menos uma seção. Use o elemento `sect`*n*. O *n* indica o número da seção, que identifica o nível da seção.

A primeira `sect`*n* é `sect1`. Você pode ter um ou mais destes em um capítulo. Eles podem conter um ou mais elementos `sect2` e assim por diante, até `sect5`.

Exemplo 9.5. Seções em Capítulos

```
<chapter>
  <title>A Sample Chapter</title>

  <para>Some text in the chapter.</para>

  <sect1>
    <title>First Section</title>

    …
  </sect1>

  <sect1>
    <title>Second Section</title>

    <sect2>
      <title>First Sub-Section</title>

      <sect3>
 <title>First Sub-Sub-Section</title>

 …
      </sect3>
    </sect2>

    <sect2>
      <title>Second Sub-Section (1.2.2)</title>

      …
    </sect2>
  </sect1>
</chapter>
```

Nota

Os números de seção são gerados automaticamente e anexados aos títulos quando o documento é renderizado em um formato de saída. Os números de seção e títulos gerados a partir do exemplo acima serão:

- 1.1. First Section
- 1.2. Second Section
- 1.2.1. First Sub-Section
- 1.2.1.1. First Sub-Sub-Section
- 1.2.2. Second Sub-Section

9.4.5. Subdividindo Utilizando Elementos `part`

`part`es adiciona outro nível de organização entre `book` e `chapter` com uma ou mais `part`es. Isso não pode ser utilizado em um `article`.

```
<part>
  <title>Introduction</title>

  <chapter>
    <title>Overview</title>

    ...
  </chapter>

  <chapter>
    <title>What is FreeBSD?</title>

    ...
  </chapter>

  <chapter>
    <title>History</title>

    ...
  </chapter>
</part>
```

9.5. Elementos Block

9.5.1. Parágrafos

O DocBook suporta três tipos de parágrafos: `formalpara`, `para` e `simpara`.

Quase todos os parágrafos da documentação do FreeBSD usam `para`. `formalpara` inclui um elemento `title`, e `simpara` desabilita alguns elementos de um `para`. Utilize mais o `para`.

Exemplo 9.6. Exemplo de um `para`

Uso:

```
<para>This is a paragraph.  It can contain just about any
  other element.</para>
```

Aparência:

This is a paragraph. It can contain just about any other element.

9.5.2. Bloco de Citações

Uma cotação em bloco é uma cotação estendida de outro documento que não deve aparecer no parágrafo atual. Estes raramente são necessários.

Os blockquotes podem, opcionalmente, conter um título e uma atribuição (ou podem ser deixados sem título e sem atribuição).

Exemplo 9.7. Exemplo `blockquote`

Uso:

```
<para>A small excerpt from the US Constitution:</para>

<blockquote>
  <title>Preamble to the Constitution of the United States</title>

  <attribution>Copied from a web site somewhere</attribution>

  <para>We the People of the United States, in Order to form a more
    perfect Union, establish Justice, insure domestic Tranquility,
    provide for the common defence, promote the general Welfare, and
    secure the Blessings of Liberty to ourselves and our Posterity, do
    ordain and establish this Constitution for the United States of
    America.</para>
</blockquote>
```

Aparência:

A small excerpt from the US Constitution:

> Preamble to the Constitution of the United States
>
> We the People of the United States, in Order to form a more perfect Union, establish Justice, insure domestic Tranquility, provide for the common defence, promote the general Welfare, and secure the Blessings of Liberty to ourselves and our Posterity, do ordain and establish this Constitution for the United States of America.
>
> —Copiado de um site qualquer

9.5.3. Dicas, Notas, Avisos, Cuidados e Informações Importantes

Informações adicionais podem precisar ser separadas do texto principal. Normalmente, essa é a informação "meta" da qual o usuário deve estar ciente.

Vários tipos de informações estão disponíveis: `tip`, `note`, `warning`, `caution`, e `important`.

O tipo a ser escolhido depende da situação. A documentação do DocBook sugere:

- A nota é para informações onde todos os leitores devem prestar atenção.

- Importante é uma variação de Nota.
- Cuidado é para informações sobre possíveis perdas de dados ou danos ao software.
- Aviso é para informações sobre possíveis danos ao hardware, sobre risco de vida ou de ferimento a um membro.

Exemplo 9.8. Example de `tip` e `important`

Uso:

```
<tip>
  <para>&os; may reduce stress.</para>
</tip>

<important>
  <para>Please use admonitions sparingly.  Too many admonitions
    are visually jarring and can have the opposite of the
    intended effect.</para>
</important>
```

Aparência:

Dica

FreeBSD may reduce stress.

Importante

Please use admonitions sparingly. Too many admonitions are visually jarring and can have the opposite of the intended effect.

9.5.4. Exemplos

Exemplos podem ser apresentados com `example`.

Exemplo 9.9. `example`

Uso:

```
<example>
  <para>Empty files can be created easily:</para>

  <screen>&prompt.user; <userinput>touch file1 file2 file3</userinput></screen>
</example>
```

Aparência:

Exemplo 9.10. `example` Renderizado

Empty files can be created easily:

```
% touch file1 file2 file3
```

9.5.5. Listas e Procedimentos

Muitas vezes, as informações precisam ser apresentadas como listas ou como uma série de etapas que devem ser realizadas para atingir um objetivo específico.

Para fazer isso, utilize `itemizedlist` , `orderedlist` , `variablelist` ou `procedure` . Existem outros tipos de elementos de lista no DocBook, mas não os cobriremos aqui.

`itemizedlist` e `orderedlist` são semelhantes às suas contrapartes em HTML, `ul` e `ol`. Cada um consiste em um ou mais elementos `listitem` , e cada `listitem` contém um ou mais elementos de bloco. Os elementos `listitem` são análogos às tags `li` do HTML. No entanto, ao contrário do HTML, eles são obrigatórios.

Exemplo 9.11. Exemplo de `itemizedlist` e `orderedlist`

Uso:

```
<itemizedlist>
  <listitem>
    <para>This is the first itemized item.</para>
  </listitem>

  <listitem>
    <para>This is the second itemized item.</para>
  </listitem>
</itemizedlist>

<orderedlist>
  <listitem>
    <para>This is the first ordered item.</para>
  </listitem>

  <listitem>
    <para>This is the second ordered item.</para>
  </listitem>
</orderedlist>
```

Aparência:

- This is the first itemized item.
- This is the second itemized item.

1. This is the first ordered item.
2. This is the second ordered item.

Uma maneira alternativa e frequentemente útil de apresentar informações é a `variablelist` . Estas são listas onde cada entrada tem um termo e uma descrição. Eles são adequados para muitos tipos de descrições e apresentam informações de uma forma que geralmente é mais fácil para o leitor do que seções e subseções.

Uma `variablelist` tem um `title` e, em seguida, pares de entradas `term` e `listitem` .

Exemplo 9.12. Exemplo `variablelist`

Uso:

```
<variablelist>
  <varlistentry>
    <term>Parallel</term>

    <listitem>
      <para>In parallel communications, groups of bits arrive
 at the same time over multiple communications
 channels.</para>
    </listitem>
  </varlistentry>

  <varlistentry>
    <term>Serial</term>

    <listitem>
      <para>In serial communications, bits arrive one at a
 time over a single communications
 channel.</para>
    </listitem>
  </varlistentry>
</variablelist>
```

Aparência:

Parallel
: In parallel communications, groups of bits arrive at the same time over multiple communications channels.

Serial
: In serial communications, bits arrive one at a time over a single communications channel.

Um `procedure` mostra uma série de `step`s, que por sua vez consistem em mais `step`s ou `substep`s. Cada `step` contém elementos de bloco e pode incluir um título opcional.

Às vezes, os passos não são sequenciais, mas apresentam uma escolha: fazer *isso* ou fazer *aquilo*, mas não ambos. Para essas escolhas alternativas, use `stepalternatives` .

Exemplo 9.13. Exemplo `procedure`

Uso:

```
<procedure>
  <step>
    <para>Do this.</para>
  </step>
```

```
  <step>
    <para>Then do this.</para>
  </step>

  <step>
    <substeps>
      <step>
        <para>And now do this smaller thing.</para>
      </step>

      <step>
        <para>And now do this other smaller thing.</para>
      </step>
    </substeps>
  </step>

  <step>
    <para>Finally, do one of these:</para>

    <stepalternatives>
      <step>
 <para>Go left.</para>
      </step>

      <step>
 <para>Go right.</para>
      </step>
    </stepalternatives>
  </step>
</procedure>
```

Aparência:

1. Do this.
2. Then do this.
3. a. And now do this small thing.

   b. And this other small thing.
4. Finally, do one of these:

   - Go left.
   - Go right.

9.5.6. Mostrando Exemplos de Arquivos

Fragmentos de um arquivo (ou talvez um arquivo completo) são mostrados agrupando-os no elemento `programlisting` .

Espaço em branco e quebra de linha dentro de `programlisting` *são* importantes. Em particular, isso significa que a tag de abertura deve aparecer na mesma linha que a primeira linha da saída, e a tag de fechamento deve aparecer na mesma linha da última linha da saída, caso contrário linhas vazias podem ser incluídas.

Exemplo 9.14. Exemplo **`programlisting`**

Uso:

```
<para>When finished, the program will look like
  this:</para>

<programlisting> #include &lt;stdio.h&gt;

int
main(void)
{
    printf("hello, world\n");
    return 0;
}</programlisting>
```

Observe como os colchetes angulares na linha `#include` precisam ser referenciados por suas entidades, em vez de serem incluídos literalmente.

Aparência:

When finished, the program will look like this:

```
#include <stdio.h>

int
main(void)
{
    printf("hello, world\n");
    return 0;
}
```

9.5.7. Chamadas

Uma chamada é um marcador visual para se referenciar a um texto ou a uma posição específica em um exemplo.

As chamadas são marcadas com o elemento `co`. Cada elemento deve ter um `id` único atribuído a ele. Após o exemplo, inclua uma `calloutlist` que descreve cada frase de destaque.

Exemplo 9.15. Exemplo `co` e `calloutlist`

```
<para>When finished, the program will look like
  this:</para>

<programlisting> #include &lt;stdio.h&gt; <co xml:id="co-ex-include"/>

int <co xml:id="co-ex-return"/>
main(void)
{
    printf("hello, world\n"); <co xml:id="co-ex-printf"/>
}</programlisting>

<calloutlist>
  <callout arearefs="co-ex-include">
    <para>Includes the standard IO header file.</para>
  </callout>

  <callout arearefs="co-ex-return">
    <para>Specifies that <function>main()</function>  returns an
      int.</para>
  </callout>

  <callout arearefs="co-ex-printf">
    <para>The <function>printf()</function>  call that writes
```

```
      <literal>hello, world</literal>  to standard output.</para>
  </callout>
</calloutlist>
```

Aparência:

When finished, the program will look like this:

```
#include <stdio.h> ❶

int ❷
main(void)
{
    printf("hello, world\n"); ❸
}
```

❶ Includes the standard IO header file.
❷ Specifies that `main()` returns an int.
❸ The `printf()` call that writes `hello, world` to standard output.

9.5.8. Tabelas

Ao contrário de HTML, o DocBook não precisa de tabelas para fins de layout, já que a folha de estilo lida com esses problemas. Em vez disso, basta usar tabelas para marcar dados tabulares.

Em termos gerais (e veja a documentação do DocBook para mais detalhes), uma tabela (que pode ser formal ou informal) consiste em um elemento `table`. Este contém pelo menos um elemento `tgroup`, que especifica (como um atributo) o número de colunas neste grupo de tabelas. Dentro do tgroup há um elemento `thead`, que contém elementos para os títulos da tabela (cabeçalhos da coluna), e um `tbody` que contém o corpo da tabela.

Ambos `tgroup` e `thead` contêm elementos `row`, que por sua vez contêm elementos `entry`. Cada elemento `entry` especifica uma célula na tabela.

Exemplo 9.16. Exemplo `informaltable`

Uso:

```
<informaltable pgwide="1">
  <tgroup cols="2">
    <thead>
      <row>
        <entry>This is Column Head 1</entry>
        <entry>This is Column Head 2</entry>
      </row>
    </thead>

    <tbody>
      <row>
 <entry>Row 1, column 1</entry>
 <entry>Row 1, column 2</entry>
      </row>

      <row>
 <entry>Row 2, column 1</entry>
 <entry>Row 2, column 2</entry>
      </row>
    </tbody>
  </tgroup>
</informaltable>
```

Aparência:

This is Column Head 1	This is Column Head 2
Row 1, column 1	Row 1, column 2
Row 2, column 1	Row 2, column 2

Sempre use o atributo `pgwide` com um valor `1` com o elemento `informaltable` . Um erro no Internet Explorer pode fazer com que a tabela seja renderizada incorretamente se isso for omitido.

As bordas da tabela podem ser escondidas configurando o atributo `frame` para `none` no elemento `informaltable` . Por exemplo, `informaltable frame="none"` .

Exemplo 9.17. Exemplo de Tabela com `frame="none"`

Aparência:

This is Column Head 1	This is Column Head 2
Row 1, column 1	Row 1, column 2
Row 2, column 1	Row 2, column 2

9.5.9. Exemplos para o Usuário Seguir

Exemplos para o usuário seguir são freqüentemente necessários. Normalmente, eles consistem em diálogos com o computador; o usuário digita um comando, o usuário recebe uma resposta de volta, o usuário digita outro comando e assim por diante.

Vários elementos e entidades podem ser utilizados nestes casos.

`screen`
: Tudo o que o usuário vê neste exemplo estará na tela do computador, então o próximo elemento é `screen` .

  Dentro da `screen` , o espaço em branco é significativo.

`prompt` , `&prompt.root;` and `&prompt.user;`
: Algumas das coisas que o usuário irá visualizar na tela são prompts do computador (seja do sistema operacional, da linha de comando shell ou de uma aplicação). Estes prompts devem ser marcados usando `prompt.`

  Por serem especiais, os prompts de shell do usuário normal e do usuário root estão disponíveis como uma entidade. Sempre que quiser indicar que o usuário está em um prompt do shell, use `&prompt.root;` para o usuário root e `&prompt.user;` para o usuário normal, conforme for necessário. Estas entidades não precisam estar dentro de um `prompt.`

Nota

`&prompt.root;` e `&prompt.user;` são extensões do FreeBSD ao DocBook e não são parte do DTD original.

`userinput`
: Ao exibir o texto que o usuário deve digitar, coloque-o nas tags `userinput` . Ele provavelmente será mostrado diferente para o usuário.

Exemplo 9.18. Exemplos `screen` , `prompt` , e `userinput`

Uso:

```
<screen> &prompt.user; <userinput> ls -1</userinput>
foo1
foo2
foo3
&prompt.user; <userinput> ls -1 | grep foo2</userinput>
foo2
&prompt.user; <userinput> su</userinput>
<prompt> Password: </prompt>
&prompt.root; <userinput> cat foo2</userinput>
This is the file called 'foo2'</screen>
```

Aparência:

```
% ls -1
foo1
foo2
foo3
% ls -1 | grep foo2
foo2
% su
Password:
# cat foo2
This is the file called 'foo2'
```

Nota

Ainda que estejamos mostrando o conteúdo do arquivo `foo2`, ele *não* está marcado como `programlisting` . Deixe o `programlisting` para mostrar fragmentos de arquivos fora do contexto de ações do usuário.

9.6. Elementos In-line

9.6.1. Realçando Informação

Para enfatizar uma palavra ou frase em particular, use `emphasis` . Isso pode ser apresentado em itálico ou negrito, ou pode ser falado diferentemente com um sistema de conversão de texto em fala.

Não há uma maneira de mudar a apresentação da ênfase no documento, não existe um equivalente a `b` e `i` do HTML . Se as informações apresentadas forem importantes, considere apresentá-las em `important` em vez de `emphasis` .

Exemplo 9.19. Exemplo de `emphasis`

Uso:

```
<para>&os; is without doubt <emphasis>the</emphasis>
  premiere &unix;-like operating system for the Intel
  architecture.</para>
```

Aparência:

FreeBSD is without doubt *the* premiere UNIX®-like operating system for the Intel architecture.

9.6.2. Siglas

Muitos termos de computador são *siglas*, palavras formadas a partir da primeira letra de cada palavra em uma frase. Os acrônimos são marcados com elementos `acronym`. É útil para o leitor quando um acrônimo é definido no primeiro uso, como mostrado no exemplo abaixo.

Exemplo 9.20. Exemplo `acronym`

Uso:

```
<para>Request For Comments (<acronym>RFC</acronym>) 1149
  defined the use of avian carriers for transmission of
  Internet Protocol (<acronym>IP</acronym>) data.  The
  quantity of <acronym>IP</acronym>  data currently
  transmitted in that manner is unknown.</para>
```

Aparência:

Request For Comments (RFC) 1149 defined the use of avian carriers for transmission of Internet Protocol (IP) data. The quantity of IP data currently transmitted in that manner is unknown.

9.6.3. Citações

Para citar texto de outro documento ou fonte, ou para denotar uma frase que é usada de forma figurada, use `quote`. A maioria das tags de marcação disponíveis para texto normal também está disponível em `quote`.

Exemplo 9.21. Exemplo `quote`

Uso:

```
<para>However, make sure that the search does not go beyond the
  <quote>boundary between local and public administration</quote>,
  as <acronym>RFC</acronym>  1535 calls it.</para>
```

Aparência:

However, make sure that the search does not go beyond the “boundary between local and public administration”, as RFC 1535 calls it.

9.6.4. Teclas, Botões do Mouse e Combinações

Para se referir a uma tecla específica no teclado, use `keycap`. Para se referir a um botão do mouse, use `mousebutton`. E para se referir a combinações de pressionamentos de teclas ou cliques do mouse, coloque todos eles em `keycombo`.

`keycombo` tem um atributo chamado `action`, que pode ser um dos `click`, `double-click` , `other`, `press`, `seq`, ou `simul`. Os dois últimos valores indicam se as teclas ou botões devem ser pressionados em sequência ou simultaneamente.

As folhas de estilo adicionam automaticamente quaisquer símbolos de conexão, como +, entre os nomes das chaves, quando agrupados em `keycombo` .

Exemplo 9.22. Exemplos de Teclas, Botões do Mouse e Combinações

Uso:

```
<para>To switch to the second virtual terminal, press
  <keycombo action="simul"> <keycap>Alt</keycap>
    <keycap>F1</keycap></keycombo> .</para>

<para>To exit <command>vi</command> without saving changes, type
  <keycombo action="seq"> <keycap>Esc</keycap><keycap>:</keycap>
    <keycap>q</keycap><keycap>!</keycap></keycombo> .</para>

<para>My window manager is configured so that
  <keycombo action="simul"> <keycap>Alt</keycap>
    <mousebutton>right</mousebutton>
  </keycombo> mouse button is used to move windows.</para>
```

Aparência:

To switch to the second virtual terminal, press Alt+F1.

To exit `vi` without saving changes, type Esc : q !.

My window manager is configured so that Alt+right mouse button is used to move windows.

9.6.5. Aplicativos, Comandos, Opções e Citações

Ambos os aplicativos e comandos são frequentemente utiilzados ao escrever uma documentação. A distinção entre eles é que um aplicativo é o nome de um programa ou conjunto de programas que preenche uma tarefa específica. Um comando é o nome do arquivo de um programa que o usuário pode digitar e executar em uma linha de comando.

Muitas vezes é necessário mostrar algumas das opções que um comando pode ter.

Finalmente, muitas vezes é útil listar um comando com seu número de seção do manual, no formato "command(number)" que é comum em manuais Unix.

Marque nomes de aplicações com `application` .

Para listar um comando com seu número de seção do manual (que deve ser utilizado com maior frequência), o elemento DocBook é `citerefentry` . Isto irá conter mais dois elementos, `refentrytitle` e `manvolnum` . O conteúdo de `refentrytitle` é o nome do comando, e o conteúdo do `manvolnum` é a seção da página de manual.

Isso pode ser trabalhoso para escrever e assim uma série de entidades gerais foram criadas para tornar esse processo mais fácil. Cada entidade assume o formato `&man.`*`manual-page`*`.`*`manual-section`*`;`.

O arquivo que contém essas entidades está em `doc/share/xml/man-refs.ent` e pode ser referenciado usando este FPI:

```
PUBLIC "-//FreeBSD//ENTITIES DocBook Manual Page Entities//EN"
```

Portanto, a introdução à documentação do FreeBSD geralmente incluirá isto:

```
<!DOCTYPE book PUBLIC "-//FreeBSD//DTD DocBook V4.1-Based Extension//EN" [

<!ENTITY % man PUBLIC "-//FreeBSD//ENTITIES DocBook Manual Page Entities//EN">
%man;

…

]>
```

Use `command` para incluir um nome de comando “in-line”, mas mostre como algo que o usuário deve digitar.

Use `option` para marcar as opções que serão passadas para um comando.

Ao se referir ao mesmo comando várias vezes nas proximidades, é preferível usar a notação `&man.command.section;` para marcação a primeira referência e use `command` para marcar as referências subsequentes. Isso faz com que a saída gerada, especialmente HTML, apareça visualmente melhor.

Exemplo 9.23. Exemplo de Aplicações, Comandos e Opções

Uso:

```
<para><application>Sendmail</application> is the most
  widely used Unix mail application.<para>

<para><application>Sendmail</application> includes the
  <citerefentry>
    <refentrytitle>sendmail</refentrytitle>
    <manvolnum>8</manvolnum>
  </citerefentry>, &man.mailq.1;, and &man.newaliases.1;
  programs.</para>

<para>One of the command line parameters to <citerefentry>
    <refentrytitle>sendmail</refentrytitle>
    <manvolnum>8</manvolnum>
  </citerefentry>, <option>-bp</option>, will display the current
  status of messages in the mail queue.  Check this on the command
  line by running <command>sendmail -bp</command>.</para>
```

Aparência:

Sendmail é o aplicativo de email Unix mais utilizado.

O Sendmail inclui o os programas sendmail(8), mailq(1), e o newaliases(1).

Um dos parâmetros da linha de comando para o sendmail(8), `-bp`, exibirá o status atual das mensagens no fila de email. Verifique isso na linha de comando executando `sendmail -bp`.

Nota

Observe como a notação `&man.section;` é mais fácil de ser seguida.

9.6.6. Arquivos, Diretórios, Extensões, Nomes de Dispositivo

Para se referir ao nome de um arquivo, um diretório, uma extensão de arquivo ou um nome de dispositivo, use `filename`.

Exemplo 9.24. Exemplo `filename`

Uso:

```
<para> O código fonte do Handbook em inglês é encontrada em
  <filename> /usr/doc/en_US.ISO8859-1/books/handbook/</filename> .
  O arquivo principal é chamado <filename> book.xml</filename> .
  Há também um <filename> Makefile</filename>  e vários arquivos com a
 extensão <filename> .ent</filename> .</para>

<para><filename> kbd0</filename>  é o primeiro teclado detectado
  pelo sistema e aparece em
  <filename> /dev</filename> . </para>
```

Aparência:

O código fonte do Handbook em inglês é encontrada em `/usr/doc/en_US.ISO8859-1/books/handbook/` . O arquivo principal é chamado `book.xml` . Há também um `Makefile` e vários arquivos com a extensão `.ent`.

`kbd0` é o primeiro teclado detectado pelo sistema e aparece em `/dev`.

9.6.7. Nome dos Ports

Extensão FreeBSD

Estes elementos fazem parte da extensão do FreeBSD ao DocBook, e não existem no DocBook DTD original.

Para incluir o nome de um programa da Coleção de Ports do FreeBSD no documento, use a tag `package`. Como a Coleção de Ports pode ser instalada em qualquer local, inclua apenas a categoria e o nome do port; não inclua `/usr/ports` .

Por padrão, `package` refere-se a um pacote binário. Para se referir a um port que será compilado a partir do código fonte, configure o atributo `role` para `port`.

Exemplo 9.25. Exemplo de `package`

Uso:

```
<para>Instale o pacote binário <package> net/wireshark</package>  para
  visualizar o tráfego de rede.</para>

<para><package role="port"> net/wireshark</package>  também pode ser
  compilado e instalado pela Coleção de Ports.</para>
```

Aparência:

Instale o pacote binário net/wireshark para visualizar o tráfego de rede.

O net/wireshark também pode ser compilado e instalado pela Coleção de Ports.

9.6.8. Hosts, Domínios, Endereços IP, Usuário, Grupo e Outros Itens do Sistema

Extensão FreeBSD

Estes elementos fazem parte da extensão do FreeBSD ao DocBook, e não existem no DocBook DTD original.

Informações para "itens do sistema" estão marcadas com `systemitem`. O atributo `class` é usado para identificar o tipo específico de informação mostrada.

`class="domainname"`
: O texto é um nome de domínio, como `FreeBSD.org` ou `ngo.org.uk`. Não há nenhum componente de nome de host.

`class="etheraddress"`
: O texto é um endereço Ethernet MAC, expresso como uma série de números hexadecimais de 2 dígitos separados por dois pontos.

`class="fqdomainname"`
: O texto é um nome de domínio FQDM, com ambos nome de domínio e hostname.

`class="ipaddress"`
: O texto é um endereço de IP.

`class="netmask"`
: O texto é uma máscara de rede, que pode ser expressa igual a um IP, uma sequência hexadecimal ou como um `/` seguido por um número (notação CIDR).

`class="systemname"`
: Com `class="systemname"`, as informações marcadas são o nome do host simples, como `freefall` ou `wcarchive`.

`class="username"`
: O texto é um nome de usuário, como `root`.

`class="groupname"`
: O texto é um grupo, como `wheel`.

Exemplo 9.26. Exemplos `systemitem`

Uso:

```
<para>The local machine can always be referred to by the
  name <systemitem class="systemname">  localhost</systemitem> , which will have the
 IP
  address <systemitem class="ipaddress">  127.0.0.1</systemitem> .</para>

<para>The <systemitem class="domainname">  FreeBSD.org</systemitem>
  domain contains a number of different hosts, including
  <systemitem class="fqdomainname">  freefall.FreeBSD.org</systemitem>  and
  <systemitem class="fqdomainname">  bento.FreeBSD.org</systemitem> .</para>

<para>When adding an <acronym> IP</acronym>  alias to an
  interface (using <command> ifconfig</command> )
  <emphasis> always</emphasis>  use a netmask of
  <systemitem class="netmask">  255.255.255.255</systemitem>  (which can
```

```
  also be expressed as
  <systemitem class="netmask">  0xffffffff</systemitem> ).</para>

<para>The <acronym> MAC</acronym>  address uniquely identifies
  every network card in existence.  A typical
  <acronym> MAC</acronym>  address looks like
  <systemitem class="etheraddress">  08:00:20:87:ef:d0</systemitem> .</para>

<para>To carry out most system administration functions
  requires logging in as <systemitem class="username">  root</systemitem> .</para>
```

Aparência:

A máquina local sempre pode ser referenciada pelo nome `localhost` , que terá o endereço IP `127.0.0.1` .

O domínio `FreeBSD.org` contém vários hosts diferentes, incluindo `freefall.FreeBSD.org` e `bento.FreeBSD.org` .

Ao adicionar um alias de IP a uma interface (usando `ifconfig`) *sempre* use uma máscara de rede de `255.255.255.255` (que também pode ser expressa como `0xffffffff`).

O endereço MAC identifica exclusivamente todas as placas de rede existentes. Um endereço MAC típico se parece com `08:00:20:87:ef:d0` .

Para executar a maioria das funções de administração do sistema, é necessário efetuar login como `root`.

9.6.9. Identificadores Uniformes de Recursos (URIs)

Ocasionalmente, é útil mostrar um Identificador de Recurso Uniforme (URI) sem torná-lo um hiperlink ativo. O elemento `uri` torna isso possível:

Exemplo 9.27. Exemplo `uri`

Uso:

```
<para>Esta URL é mostrada apenas como texto:
  <uri>https://www.FreeBSD.org</uri>. Não é criado um link.</para>
```

Aparência:

Esta URL é mostrada apenas como texto: `https://www.FreeBSD.org` . Não é criado um link.

Para criar links, consulte Seção 9.8, “Links”.

9.6.10. Endereços de Email

Os endereços de email são marcados como elementos `email`. No formato de saída HTML, o texto marcado se torna um hiperlink para o endereço de email. Outros formatos de saída que suportam hiperlinks também podem tornar o endereço de email em um link.

Exemplo 9.28. Exemplo de Hiperlink com `email`

Uso:

```
<para> Um endereço de email que não existe, como
  <email> notreal@example.com</email> , pode ser usado como um
  exemplo </para>
```

Aparência:

Um endereço de email que não existe, como <notreal@example.com >, pode ser usado como exemplo.

Uma extensão específica do FreeBSD permite configurar o atributo `role` para `nolink` para evitar a criação do hiperlink para o endereço de email.

Exemplo 9.29. Exemplo de `email` Sem Hiperlink

Uso:

```
<para> Às vezes, o link para um endereço de email como
  <email role="nolink"> notreal@example.com</email> não é
  desejado.</para>
```

Aparência:

Às vezes, o link para um endereço de email como <notreal@example.com > não é desejado.

9.6.11. Descrevendo `Makefile`s

Extensão FreeBSD

Estes elementos fazem parte da extensão do FreeBSD ao DocBook, e não existem no DocBook DTD original.

Existem dois elementos para descrever partes de `Makefile`s, `buildtarget` e `varname`.

`buildtarget` identifica um destino de compilação exportado por um `Makefile` que pode ser fornecido como um parâmetro para o `make`. `varname` identifica uma variável que pode ser definida (no ambiente, na linha de comando com o `make`, ou dentro do `Makefile`) para influenciar o processo.

Exemplo 9.30. Exemplo de `buildtarget` e `varname`

Uso:

```
<para>Two common targets in a <filename> Makefile</filename>
  are <buildtarget> all</buildtarget>  and
  <buildtarget> clean</buildtarget> .</para>

<para>Typically, invoking <buildtarget> all</buildtarget>  will
  rebuild the application, and invoking
  <buildtarget> clean</buildtarget>  will remove the temporary
  files (<filename> .o</filename>  for example) created by the
  build process.</para>

<para><buildtarget> clean</buildtarget>  may be controlled by a
```

```
  number of variables, including <varname> CLOBBER</varname>
  and <varname> RECURSE</varname> .</para>
```

Aparência:

Dois targets comuns em um `Makefile` são `all` e `clean`.

Normalmente, invocar `all` irá reconstruir o aplicativo, e invocar `clean` removerá os arquivos temporários (`.o` por exemplo) criados pelo processo de compilação.

`clean` pode ser controlado por várias de variáveis, incluindo `CLOBBER` e `RECURSE`.

9.6.12. Texto Literal

O texto literal, ou texto que deve ser inserido na íntegra, é frequentemente necessário na documentação. Este é um texto extraído de outro arquivo ou que deve ser copiado exatamente como mostrado na documentação em um outro arquivo.

Na maioria das vezes, `programlisting` será suficiente para denotar este texto. Mas `programlisting` nem sempre é apropriado, particularmente quando você quer incluir uma parte de um arquivo "in-line" com o resto do parágrafo.

Nestes casos, use `literal`.

Exemplo 9.31. Exemplo `literal`

Uso:

```
<para>The <literal> maxusers 10</literal>  line in the kernel
  configuration file determines the size of many system tables, and is
  a rough guide to how many simultaneous logins the system will
  support.</para>
```

Aparência:

A linha `maxusers 10` no arquivo de configuração do kernel determina o tamanho de muitas tabelas do sistema e é um guia aproximado de quantos logins simultâneos o sistema suportará.

9.6.13. Mostrando Itens que o Usuário Deve Preencher

Haverá diversos momentos em que o usuário irá ver o que fazer, ou será referenciado a um arquivo ou linha de comando, mas não poderá simplesmente copiar o exemplo fornecido. Em vez disso, eles precisam fornecer algumas informações.

`replaceable` é projetado para essa eventualidade. Use isso *dentro* de outros elementos para indicar partes do conteúdo desse elemento que o usuário deve substituir.

Exemplo 9.32. Exemplo de `replaceable`

Uso:

```
<screen> &prompt.user; <userinput> man <replaceable> command</replaceable> </
userinput> </screen>
```

Aparência:

```
% man command
```

`replaceable` pode ser usado em diversos elementos, incluindo `literal`. Este exemplo também mostra que o `replaceable` deve estar apenas em torno do conteúdo que o usuário *está* destinado a fornecer. O outro conteúdo deve ser deixado de lado.

Uso:

```
<para>The <literal>maxusers <replaceable>n</replaceable></literal>
  line in the kernel configuration file determines the size of many system
  tables, and is a rough guide to how many simultaneous logins the system will
  support.</para>

<para>For a desktop workstation, <literal>32</literal> is a good value
  for <replaceable>n</replaceable>.</para>
```

Aparência:

A linha `maxusers` *n* no arquivo de configuração do kernel determina o tamanho de muitas tabelas do sistema e é um guia aproximado de quantos logins simultâneos o sistema suportará.

Para uma estação de trabalho, `32` é um bom valor para *n*.

9.6.14. Mostrando Botões GUI

Os botões apresentados por uma interface gráfica do usuário são marcados com `guibutton`. Para tornar o texto mais parecido com um botão gráfico, colchetes e espaços não separáveis são adicionados em volta do texto.

Exemplo 9.33. Exemplo `guibutton`

Uso:

```
<para>Edit the file, then click
  <guibutton>[ Save ]</guibutton> to save the
  changes.</para>
```

Aparência:

Edite o arquivo e clique em [Salvar] para salvar as alterações.

9.6.15. Citações de Erros do Sistema

Erros de sistema gerados pelo FreeBSD são marcados com `errorname`. Isso indica o erro exato que aparece.

Exemplo 9.34. Exemplo `errorname`

Uso:

```
<screen><errorname>Panic: cannot mount root</errorname></screen>
```

Aparência:

```
Panic: cannot mount root
```

9.7. Imagens

Importante

O suporte de imagem na documentação é um pouco experimental. Os mecanismos descritos aqui provavelmente não mudarão, mas isso não é garantido.

Para fornecer conversão entre diferentes formatos de imagem, o port graphics/ImageMagick deve estar instalado. Esse port não está incluído no meta port textproc/docproj e deve ser instalado separadamente.

Um bom exemplo do uso de imagens é o documento `graphics/ImageMagick .` Examine os arquivos nesse diretório para ver como esses elementos são usados juntos. Crie formatos de saída diferentes para ver como o formato determina quais imagens são mostradas no documento renderizado.

9.7.1. Formatos de Imagem

Os seguintes formatos de imagem são suportados atualmente. Um arquivo de imagem será automaticamente convertido em bitmap ou imagem vetorial, dependendo do formato do documento de saída.

Estes são os formatos *somente* nos quais as imagens devem ser enviadas para o repositório de documentação.

EPS (Encapsulated Postscript)
: Imagens com base principalmente em vetores, como diagramas de rede, linhas de tempo e similares, devem estar nesse formato. Estas imagens têm uma extensão `.eps`.

PNG (Portable Network Graphic)
: Para bitmaps, como capturas de tela, use este formato. Essas imagens têm a extensão `.png`.

PIC (PIC graphics language)
: PIC é uma linguagem para desenhar figuras baseadas em vetor simples usadas no utilitário pic(1). Essas imagens têm a extensão `.pic`.

SCR (SCReen capture)
: Este formato é específico para capturas de tela da saída do console. O seguinte comando gera um arquivo SCR `shot.scr` do buffer de vídeo `/dev/ttyv0` :

  ```
  # vidcontrol -p  < /dev/ttyv0  > shot.scr
  ```

  É preferível o formato PNG para capturas de tela porque o arquivo SCR contém texto sem formatação das linhas de comando para que possa ser convertido em uma imagem PNG ou um texto simples, dependendo do formato do documento de saída.

Use o formato apropriado para cada imagem. A documentação geralmente tem uma mistura de imagens EPS e PNG. O `Makefile` assegura que a imagem de formato correta seja escolhida dependendo do formato de saída usado. *Não envie a mesma imagem ao repositório em dois formatos diferentes.*

Importante

O Projeto de Documentação pode eventualmente mudar para o formato SVG (Scalable Vector Graphic) para imagens vetoriais. No entanto, o estado atual das ferramentas de edição compatíveis com SVG torna isso inviável.

9.7.2. Localizações das Imagens

As imagens podem ser armazenados em um dos diversos locais abaixo, dependendo do documento e da imagem:

- No mesmo diretório do documento em si, geralmente feito para artigos e pequenos livros que mantêm todos os arquivos em um único diretório.
- Em um subdiretório do documento principal. Geralmente feito quando um livro grande usa subdiretórios separados para organizar capítulos individuais.

  Quando as imagens são armazenadas em um subdiretório do diretório do documento principal, o nome do subdiretório deve ser incluído em seus caminhos no `Makefile` e no elemento `imagedata`.
- Em um subdiretório de `doc/share/images` nomeado após o documento. Por exemplo, as imagens do Handbook estão armazenadas em `doc/share/images/books/handbook`. Imagens que funcionam para várias traduções são armazenadas neste nível superior da árvore de arquivos da documentação. Geralmente, estas são imagens que podem ser usadas inalteradas em traduções não inglesas do documento.

9.7.3. Marcação em Imagem

As imagens são incluídas como parte de um `mediaobject`. O `mediaobject` pode conter outros objetos mais específicos. Estamos preocupados com dois, o `imageobject` e o `textobject`.

Inclua um `imageobject` e dois elementos `textobject`. O `imageobject` apontará para o nome do arquivo de imagem sem a extensão. Os elementos `textobject` contêm informações que serão apresentadas ao usuário, bem como, ou em vez da própria imagem.

Elementos de texto são mostrados ao leitor em várias situações. Quando o documento é exibido em HTML, elementos de texto são mostrados enquanto a imagem está sendo carregada ou se o ponteiro do mouse estiver sobre a imagem ou se um navegador somente texto estiver sendo usado. Em formatos como texto simples, onde os gráficos não são possíveis, os elementos de texto são mostrados em vez dos gráficos.

Este exemplo mostra como incluir uma imagem chamada `fig1.png` em um documento. A imagem é um retângulo com um A dentro dela:

```
<mediaobject>
  <imageobject>
    <imagedata fileref="fig1"/>   ❶
  </imageobject>

  <textobject>
    <literallayout class="monospaced">  +---------------+ ❷
|       A       |
+---------------+</literallayout>
  </textobject>

  <textobject>
    <phrase>A picture</phrase>  ❸
  </textobject>
</mediaobject>
```

❶ Inclua um elemento imagedata dentro do elemento imageobject. O atributo fileref deve conter o nome do arquivo da imagem a ser incluída, sem a extensão. As folhas de estilo irão descobrir qual extensão deve ser adicionada ao nome do arquivo automaticamente.

❷ O primeiro textobject contém um elemento literallayout, onde o atributo class é definido como monospaced. Esta é uma oportunidade para demonstrar habilidades artísticas em ASCII. Este conteúdo será usado se o documento for convertido em texto simples.

Observe como a primeira e a última linha do conteúdo do elemento literallayout aparecem ao lado das tags do elemento. Isso garante que nenhum espaço em branco seja incluído.

❸ O segundo textobject contém um único elemento phrase. O conteúdo desta frase se tornará o atributo alt para a imagem quando este documento for convertido para HTML.

9.7.4. Entradas de Imagem no Makefile

As imagens devem estar listadas no Makefile na variável IMAGES. Esta variável deve conter os nomes de todas as imagens *source*. Por exemplo, se houver três figuras, fig1.eps, fig2.png, fig3.png, então o Makefile deve ter linhas como esta.

```
…
IMAGES= fig1.eps fig2.png fig3.png
…
```

ou

```
…
IMAGES=  fig1.eps
IMAGES+= fig2.png
IMAGES+= fig3.png
…
```

Novamente, o Makefile irá elaborar a lista completa de imagens necessárias para compilar o documento, você só precisa listar os arquivos de imagem que *você* forneceu.

9.7.5. Imagens e Capítulos em Subdiretórios

Tenha cuidado ao separar a documentação em arquivos menores em diretórios diferentes (consulte Seção 7.7.1, “Usando Entidades Gerais para Incluir Arquivos”).

Imagine que haja um livro com três capítulos e os capítulos sejam armazenados em seus próprios diretórios, chamados chapter1/chapter.xml, chapter2/chapter.xml e chapter3/chapter.xml. Se cada capítulo tiver imagens associadas, coloque essas imagens no subdiretório de cada capítulo (chapter1/, chapter2/, e chapter3/).

No entanto, fazer isso requer a inclusão dos nomes de diretório na variável IMAGES no Makefile, *e* a inclusão do nome do diretório no elemento imagedata no documento.

Por exemplo, se o livro tiver chapter1/fig1.png, então chapter1/chapter.xml deve conter:

```
<mediaobject>
  <imageobject>
    <imagedata fileref="chapter1/fig1"/>    ❶
  </imageobject>

  …

</mediaobject>
```

❶ O nome do diretório deve ser incluído no atributo fileref.

O Makefile deve conter:

```
…
IMAGES=  chapter1/fig1.png
```

...

9.8. Links

Nota

Links também são elementos in-line. Para mostrar uma URI sem criar um link, consulte Seção 9.6.9, “Identificadores Uniformes de Recursos (URIs)”.

9.8.1. Atributos `xml:id`

A maioria dos elementos DocBook aceita um atributo `xml:id` para dar a essa parte do documento um nome exclusivo. O `xml:id` pode ser usado como um destino para uma referência cruzada ou link.

Qualquer parte do documento que será um destino de link deve ter um atributo `xml:id`. Atribuir um `xml:id` a todos os capítulos e seções, mesmo que não haja planos atuais para criar link para eles, é uma boa ideia. Esses `xml:id` s podem ser usados como pontos de referência exclusivos por qualquer pessoa que se refira à versão do documento HTML.

Exemplo 9.35. `xml:id` em Ccapítulos e sSeções de eExemplo

```
<chapter xml:id="introduction">
  <title>Introduction</title>

  <para>This is the introduction.  It contains a subsection,
    which is identified as well.</para>

  <sect1 xml:id="introduction-moredetails">
    <title>More Details</title>

    <para>This is a subsection.</para>
  </sect1>
</chapter>
```

Use valores descritivos para nomes de `xml:id`. Os valores devem ser exclusivos em todo o documento, não apenas em um único arquivo. No exemplo, a subseção `xml:id` é construída anexando o texto ao capítulo `xml:id`. Isso garante que os `xml:id` s sejam exclusivos. Ele também ajuda o leitor e qualquer um que editar o documento a ver onde o link está localizado no documento, semelhante a um caminho de diretório para um arquivo.

9.8.2. Referências Cruzadas com `xref`

`xref` fornece ao leitor um link para ir para outra seção do documento. O destino `xml:id` é especificado no atributo `linkend` e `xref` gera o texto do link automaticamente.

Exemplo 9.36. Exemplo `xref`

Imagine que esse fragmento apareça em algum lugar em um documento que inclua o exemplo `xml:id` mostrado acima:

```
<para>More information can be found
```

```
  in <xref linkend="introduction"/>  .</para>

<para>More specific information can be found
  in <xref linkend="introduction-moredetails"/>   .</para>
```

O texto do link será gerado automaticamente, parecendo como (O *enfatizado* indica o texto do link):

> Mais informações podem ser encontradas no *Capítulo 1, Introdução*.
>
> Informações mais específicas podem ser encontradas em *Seção 1.1, "Mais Detalhes"*.

O texto do link é gerado automaticamente a partir do capítulo e número da seção e elementos `title`.

9.8.3. Criando Links para Outros Documentos na Web

O elemento link descrito aqui permite que o escritor defina o texto do link. Quando o texto do link é usado, é muito importante ser descritivo para dar ao leitor uma ideia de onde o link vai. Lembre-se que o DocBook pode ser renderizado para vários tipos de mídia. O leitor pode estar olhando para um livro impresso ou outra forma de mídia onde não há links. Se o texto do link não for descritivo o suficiente, talvez o leitor não consiga localizar a seção vinculada.

O atributo `xlink:href` é a URL da página, e o conteúdo do elemento é o texto que será exibido para o usuário ativar.

Em muitas situações, é preferível mostrar a URL real em vez do texto. Isso pode ser feito deixando de fora o texto do elemento.

Exemplo 9.37. `link` para um Exemplo de Página Web de Documentação do FreeBSD

Link para uma URL de uma entidade. de livro ou artigo Para criar um link para um capítulo específico de um livro, adicione uma barra e o nome do arquivo do capítulo, seguido por uma âncora opcional dentro do capítulo. Para artigos, crie um link para a entidade URL do artigo, seguida por uma âncora opcional no artigo. As entidades URL podem ser encontradas em `doc/share/xml/urls.ent` .

Uso para links de livros do FreeBSD:

```
<para>Read the <link
    xlink:href="&url.books.handbook;/svn.html#svn-intro">   SVN
    introduction</link>, then pick the nearest mirror from
  the list of <link
    xlink:href="&url.books.handbook;/svn.html#svn-mirrors">   Subversion
    mirror sites</link>.</para>
```

Aparência:

Leia a Introdução ao SVN, em seguida, escolha o espelho mais próximo do lista de sites espelho do Subversion.

Uso para links de artigos do FreeBSD:

```
<para>Read this
  <link xlink:href="&url.articles.bsdl-gpl;">  article
    about the BSD license</link>, or just the
  <link xlink:href="&url.articles.bsdl-gpl;#intro">   introduction</link>.</para>
```

Aparência:

Leia este artigo sobre a licença BSD, ou apenas a sua introdução.

Exemplo 9.38. `link` para um Exemplo de Página Web do FreeBSD

Uso:

```
<para>Of course, you could stop reading this document and go to the
  <link xlink:href="&url.base;/index.html">  FreeBSD home page</link> instead.</para>
```

Aparência:

Claro, você pode parar de ler este documento e ir para a página inicial do FreeBSD.

Exemplo 9.39. `link` para um Exemplo de Página Web Externa

Uso:

```
<para>Wikipedia has an excellent reference on
  <link
    xlink:href="http://en.wikipedia.org/wiki/GUID_Partition_Table">   GUID
    Partition Tables</link> .</para>
```

Aparência:

A Wikipedia tem uma excelente referência em Tabelas de Partição GUID.

O texto do link pode ser omitido para mostrar a URL real:

```
<para>Wikipedia has an excellent reference on
  GUID Partition Tables: <link
    xlink:href="http://en.wikipedia.org/wiki/GUID_Partition_Table">   </link> .</para>
```

O mesmo link pode ser inserido usando notação mais curta em vez de uma tag de finalização separada:

```
<para>Wikipedia has an excellent reference on
  GUID Partition Tables: <link
    xlink:href="http://en.wikipedia.org/wiki/GUID_Partition_Table"/>   .</para>
```

Os dois métodos são equivalentes. Aparência:

A Wikipedia tem uma excelente referência em Tabelas de Partição GUID: `http://en.wikipedia.org/wiki/GUID_Partition_Table` .

Capítulo 10. Folhas de Estilo

O XML se preocupa com o conteúdo e não diz nada sobre como esse conteúdo deve ser apresentado ao leitor ou renderizado no papel. Múltiplas linguagens de *folha de estilo* foram desenvolvidas para descrever o layout visual, incluindo a Transformação de Linguagem de Folha de Estilo Extensível (XSLT), Semântica de Estilo de Documento e Linguagem de Especificação (DSSSL) e Folhas de Estilo em Cascata (CSS).

Os documentos do FDP usam folhas de estilo XSLT para transformar o DocBook em XHTML, e então a formatação CSS é aplicada nas páginas XHTML. A saída imprimível é atualmente renderizada com folhas de estilo legadas do DSSSL, mas isso provavelmente mudará no futuro.

10.1. CSS

Folhas de estilos em cascata (CSS) são um mecanismo para anexar informações de estilo (font, weight, size, color e assim por diante) a elementos em um documento XHTML sem abusar de XHTML para o fazer.

10.1.1. Os Documentos DocBook

As folhas de estilo XSLT e DSSSL do FreeBSD se referem a `docbook.css` , que deve estar presente no mesmo diretório que os arquivos XHTML. O arquivo CSS de todo o projeto é copiado de `doc/share/misc/docbook.css` quando os documentos são convertidos para XHTML e são instalados automaticamente .

Capítulo 11. Traduções

Este é o FAQ para pessoas que estão traduzindo a documentação do FreeBSD (FAQ, Handbook, tutoriais, páginas de manual e outros) para diferentes idiomas.

Ele é *fortemente* baseado na tradução do FAQ do Projeto de Documentação Alemão do FreeBSD, originalmente escrito por Frank Gründer <elwood@mc5sys.in-berlin.de > e traduzido de volta para o inglês por Bernd Warken <bwarken@mayn.de >.

O FAQ é mantido pela Equipe de Engenharia de Documentação <doceng@FreeBSD.org >.

P: O que significa i18n e l10n?

R: i18n significa internacionalização e l10n significa localização. São apenas uma abreviação.

i18n pode ser lido como “i” seguido por 18 letras, seguido por “n”. Da mesma forma, l10n é “l” seguido por 10 letras, seguido por “n”.

P: Existe uma lista de discussão para tradutores?

R: Sim. Diferentes grupos de tradução têm suas próprias listas de discussão. A lista dos projetos de tradução possui mais informações sobre as listas de discussão e sites web mantidos por cada projeto de tradução. Além disso, existe a <freebsd-translators@freebsd.org > para discussão geral de tradução.

P: São necessários mais tradutores?

R: Sim. Quanto mais pessoas trabalham na tradução, mais rápido ela será finalizada, e mais rapidamente as mudanças na documentação em Inglês serão refletidas nos documentos traduzidos.

Você não precisa ser um tradutor profissional para poder ajudar.

P: Quais idiomas eu preciso saber?

R: Idealmente, você deverá possuir um bom conhecimento de Inglês escrito, e obviamente, precisará ser fluente no idioma para o qual está traduzindo.

Inglês não é estritamente necessário. Por exemplo, você poderia fazer uma tradução Húngara do FAQ da tradução em Espanhol.

P: Quais softwares eu preciso conhecer?

R: É fortemente recomendado que você mantenha uma cópia local do repositório Subversion do FreeBSD (pelo menos a parte da documentação). Isso pode ser feito executando:

```
% svn checkout https://svn.FreeBSD.org/doc/head/ head
```

svn.FreeBSD.org é um servidor público SVN. Verifique o certificado do servidor na lista de Mirrors de Subversion.

Nota

Será necessário que o pacote devel/subversion esteja instalado.

Você deverá ter conhecimentos básicos de svn. Ele permitirá que você veja o que mudou entre as diferentes versões dos arquivos que compõem a documentação.

Por exemplo, para ver as diferenças entre as revisões r33733 e r33734 do en_US.ISO8859-1/books/fdp-primer/book.xml , execute:

```
% svn diff -r 33733:33734 en_US.ISO8859-1/books/fdp-primer/book.xml
```

Por favor, veja a explicação completa de como usar o Subversion no FreeBSD no FreeBSD Handbook.

P: Como eu faço para descobrir se já existem outras pessoas traduzindo documentos para o meu idioma?

R: A página do Projeto de Tradução da Documentação lista os trabalhos de tradução que são conhecidos. Se outras pessoas já estiverem trabalhando na tradução de documentação para o seu idioma, por favor, não duplique os esforços. Em vez disso, entre em contato para saber como você pode ajudar.

Se não existir nenhum projeto de tradução para o seu idioma listado nesta página, envie uma mensagem para lista de discussão do projeto de documentação do FreeBSD para o caso de alguém estar pensando em fazer a tradução, mas ainda não tiver anunciado nada.

P: Ninguém mais está traduzindo para o meu idioma. O que eu faço?

R: Parabéns, você acabou de iniciar o " Projeto de Tradução da Documentação do FreeBSD em *`seu idioma aqui`* ". Bem vindo a bordo.

Primeiro, pense se você terá o tempo necessário. Uma vez que você é a única pessoa trabalhando no seu idioma no momento, será sua a responsabilidade de publicar o seu trabalho e coordenar qualquer voluntário que queira ajudá-lo.

Escreva um email para a lista de discussão do Projeto de Documentação, anunciando que você irá traduzir a documentação, assim a página do Projeto de Traduções de Documentação poderá ser atualizada.

Se já existir alguém em seu país provendo o espelhamento de serviços do FreeBSD, você deve contacta-lo e perguntar se você pode ter algum espaço web para seu projeto, e se possível um endereço de email ou mesmo um serviço de lista de discussão.

Então escolha um documento e comece a traduzir. É melhor começar com algo razoavelmente pequeno—como o FAQ ou um dos tutoriais.

P: Eu já tenho alguns documentos traduzidos, para onde eu devo enviá-los?

R: Isso depende. Se você já está trabalhando com uma equipe de tradução (tal como a equipe japonesa, ou a equipe alemã) então ela terá seus próprios procedimentos para manipular a documentação submetida, e estes serão descritos em seus web sites.

Se você for a única pessoa trabalhando em um determinado idioma (ou se você é o responsável pelo projeto de tradução e quer submeter suas mudanças de volta para o projeto FreeBSD) então você deve enviar sua tradução ao Projeto FreBSD (veja pergunta seguinte).

P: Eu sou a única pessoa trabalhando na tradução para este idioma, como faço para enviar minha tradução?

ou

Somos uma equipe de tradução e queremos enviar a documentação que nossos membros traduziram para nós.

R: Primeiro, verifique se sua tradução está organizada corretamente. Isso significa que ele deve cair na árvore de documentação existente e ser compilada de maneira correta.

Atualmente a documentação do FreeBSD é armazenada em um diretório de nível superior chamado `head/`. Os diretórios abaixo desse são nomeados de acordo com o código do idioma em que estão escritos, conforme definido na ISO639 (`/usr/share/misc/iso639` em uma versão do FreeBSD mais recente que 20 de janeiro de 1999).

Se o seu idioma puder ser codificado de maneiras diferentes (por exemplo, Chinês), deve haver diretórios abaixo desse, um para cada formato de codificação fornecido.

Finalmente, você deve ter diretórios para cada documento.

Por exemplo, em uma hipotética tradução Sueca ficaria assim:

```
head/
    sv_SE.ISO8859-1/
                    Makefile
                    htdocs/
                          docproj/
                    books/
                          faq/
                              Makefile
                              book.xml
```

`sv_SE.ISO8859-1` é o nome da tradução, no formato *`lang.encoding`*. Repare nos dois Makefiles, que serão usados para compilar a documentação.

Use tar(1) e gzip(1) para compactar sua documentação e enviá-lo para o projeto.

```
% cd doc
% tar cf swedish-docs.tar sv_SE.ISO8859-1
% gzip -9 swedish-docs.tar
```

Coloque `swedish-docs.tar.gz` em algum lugar. Se você não tiver acesso ao seu próprio espaço web (talvez seu ISP não disponibilize um), envie um email para a Equipe de Engenharia da Documentação <`doceng@FreeBSD.org`> e combine o envio dos arquivos por email conveniente quando for conveniente.

De qualquer maneira, você deve usar o Bugzilla para enviar um relatório indicando que você enviou a documentação. Seria muito útil se você conseguir outras pessoas para checar sua tradução primeiro, já que é improvável que a pessoa que irá fazer o commit seja fluente no idioma.

Alguém (provavelmente o Gerente do Projeto de Documentação, atualmente Equipe de Engenharia de Documentação <`doceng@FreeBSD.org`>) irá então pegar sua tradução e checar se ela compila. Em particular, as seguintes coisas serão analisadas:

1. Todos os seus arquivos usam strings RCS (tais como "ID")?

2. O `make all` no diretório `sv_SE.ISO8859-1` funciona corretamente?

3. O `make install` funciona corretamente?

Se houver algum problema, quem estiver validando a submissão irá entrar em contato para que seja feito as correções.

Se não houver problemas, sua tradução será disponibilizada o mais rápido possível.

P: Posso incluir um texto específico do idioma ou do país em minha tradução?

R: Nós preferimos que você não faça isso.

Por exemplo, suponha que você esteja traduzindo o Handbook para o Coreano e queira incluir uma seção sobre varejistas na Coreia em seu Handbook.

Não há razão pela qual esta informação não deva estar nas versões em Inglês (ou Alemão, ou Espanhol, ou Japonâs, ou ...). É possível que uma pessoa que fale Inglês na Coréia possa tentar obter uma cópia do FreeBSD enquanto esteja ali. Isso também ajuda a aumentar a presença perceptível do FreeBSD ao redor do mundo, o que não é uma coisa ruim.

Se você tiver informações específicas do país, submeta-as como uma alteração do Handbook em Inglês (usando o Bugzilla) e em seguida, traduza a alteração de volta para o seu idioma no Handbook traduzido.

Obrigado.

P: Como os caracteres específicos do idioma podem ser incluídos?

R: Caracteres não-ASCII na documentação devem ser incluídos usando entidades SGML.

Resumidamente, eles se parecem com um "e" comercial (&), o nome da entidade e um ponto e vírgula (;).

Os nomes das entidades são definidos em ISO8879, que está na árvore de ports em textproc/iso8879.

Alguns exemplos incluem:
Entidade: é
Aparência: é
Descrição: “e” minúsculo com acento agudo
Entidade: É
Aparência: É
Descrição: “E” maiúsculo com acento agudo
Entidade: ü
Aparência: ü
Descrição: “u” minúsculo com trema

Depois de instalar o port iso8879, os arquivos em `/usr/local/share/xml/iso8879` conterão a lista completa.

P: Dirigindo-se ao leitor

R: Nos documentos em Inglês, o leitor é tratado por “você”, não há distinção entre formal/informal como existe em alguns idiomas.

Se você estiver traduzindo para um idioma que tenha esta distinção, use qualquer forma que normalmente é usada em outras documentações técnicas. Na dúvida, use a forma mais educada.

P: Preciso incluir informações adicionais nas minhas traduções?

R: Sim.

O cabeçalho da versão em Inglês de cada documento será algo parecido com isto:

```
<!--
     The FreeBSD Documentation Project

     $FreeBSD$
-->
```

O forma exata pode mudar, mas sempre incluirá uma linha $FreeBSD$ e a frase `The FreeBSD Documentation Project`. Note que a parte do $FreeBSD é expandida automaticamente pelo Subversion, portanto ela deve estar vazia (apenas `$FreeBSD$`) para novos arquivos.

Seus documentos traduzidos devem incluir sua própria linha $FreeBSD$, e mudar a linha `FreeBSD Documentation Project` para `The FreeBSD` *`language`* `Documentation Project .`

Você deve ainda adicionar uma terceira linha que indicará qual revisão do texto em inglês o texto é baseado.

Assim, a versão em Espanhol desse arquivo pode começar com:

```
<!--
     The FreeBSD Spanish Documentation Project

     $FreeBSD$
     Original revision: r38674
-->
```

Capítulo 12. Traduções PO

12.1. Introdução

O GNU gettext oferece aos tradutores uma maneira fácil de criar e manter traduções de documentos. Sequências traduzíveis são extraídas do documento original em um arquivo PO (Portable Object). Versões traduzidas das strings são inseridas com um editor separado. As strings podem ser usadas diretamente ou incorporadas em uma versão traduzida completa do documento original.

12.2. Quick Start

Supõe-se que o procedimento mostrado em Seção 1.1, "Quick Start" já tenha sido executado. A opção `TRANSLATOR` é necessária e já está ativada por padrão no port textproc/docproj.

Este exemplo mostra a criação de uma tradução em Espanhol do pequeno artigo Leap Seconds.

Procedimento 12.1. Instale um Editor PO

- É necessário um editor PO para editar os arquivos de tradução. Este exemplo utiliza o editors/poedit.

```
# cd /usr/ports/editors/poedit
# make install clean
```

Procedimento 12.2. Configuração Inicial

Quando uma nova tradução é criada pela primeira vez, a estrutura do diretório e o `Makefile` devem ser criados ou copiados do original em Inglês:

1. Crie um diretório para a nova tradução. O código fonte do artigo em Inglês está em `~/doc/en_US.ISO8859-1/articles/leap-seconds/` . A tradução em Espanhol estará em `~/doc/es_ES.ISO8859-1/articles/leap-seconds/` . O caminho é o mesmo, exceto pelo nome do diretório de idiomas.

```
% svn mkdir --parents ~/doc/es_ES.ISO8859-1/articles/leap-seconds/
```

2. Copie o `Makefile` do documento original para o diretório de tradução:

```
% svn cp ~/doc/en_US.ISO8859-1/articles/leap-seconds/Makefile \
    ~/doc/es_ES.ISO8859-1/articles/leap-seconds/
```

Procedimento 12.3. Tradução

A tradução de um documento consiste em duas etapas: extrair strings traduzíveis do documento original e inserir as traduções dessas strings. Essas etapas são repetidas até que o tradutor sinta que o documento foi traduzido o suficiente para produzir um documento traduzido que seja utilizável.

1. Extraia as strings traduzíveis da versão original em Inglês para um arquivo PO:

```
% cd ~/doc/es_ES.ISO8859-1/articles/leap-seconds/
% make po
```

2. Use um editor PO para inserir as traduções no arquivo PO. Existem vários editores diferentes disponíveis. O `poedit` do editors/poedit é mostrado aqui.

   O nome do arquivo PO é o código de idioma de dois caracteres seguido por um código de região de também dois caracteres, seprados por um underline. Para espanhol, o nome do arquivo é `es_ES.po` .

```
% poedit es_ES.po
```

Procedimento 12.4. Gerando um Documento Traduzido

1. Gere o documento traduzido:

```
% cd ~/doc/es_ES.ISO8859-1/articles/leap-seconds/
% make tran
```

O nome do documento gerado corresponde ao nome do original em Inglês, geralmente `article.xml` para artigos ou `book.xml` para livros.

2. Verifique o arquivo gerado renderizando-o para HTML e exibindo-o com um navegador web:

```
% make FORMATS=html
% firefox article.html
```

12.3. Criando Novas Traduções

O primeiro passo para criar um novo documento traduzido é localizar ou criar um diretório para mantê-lo. O FreeBSD coloca documentos traduzidos em um subdiretório nomeado para seu idioma e região no formato *`lang_REGION`* . *`lang`* é um código minúsculo de dois caracteres. Ele é seguido por um caractere underline, em seguida, pelo código de duas letras maiúsculas *`REGION`*.

Tabela 12.1. Nomes de Idioma

Língua	Região	Nome do Diretório Traduzido	Nome do Arquivo PO	Conjunto de Caracteres
Inglês	Estados Unidos	`en_US.ISO8859-1`	`en_US.po`	ISO 8859-1
Bengali	Bangladesh	`bn_BD.UTF-8`	`bn_BD.po`	UTF-8
Dinamarquês	Dinamarca	`da_DK.ISO8859-1`	`da_DK.po`	ISO 8859-1
Alemão	Alemanha	`de_DE.ISO8859-1`	`de_DE.po`	ISO 8859-1
Grego	Grécia	`el_GR.ISO8859-7`	`el_GR.po`	ISO 8859-7
Espanhol	Espanha	`es_ES.ISO8859-1`	`es_ES.po`	ISO 8859-1
Francês	França	`fr_FR.ISO8859-1`	`fr_FR.po`	ISO 8859-1
Húngaro	Hungria	`hu_HU.ISO8859-2`	`hu_HU.po`	ISO 8859-2
Italiano	Itália	`it_IT.ISO8859-15`	`it_IT.po`	ISO 8859-15
Japonês	Japão	`ja_JP.eucJP`	`ja_JP.po`	EUC JP
Coreano	Coréia	`ko_KR.UTF-8`	`ko_KR.po`	UTF-8
Mongol	Mongólia	`mn_MN.UTF-8`	`mn_MN.po`	UTF-8
Holandês	Holanda	`nl_NL.ISO8859-1`	`nl_NL.po`	ISO 8859-1
Polonês	Polônia	`pl_PL.ISO8859-2`	`pl_PL.po`	ISO 8859-2
Português	Brasil	`pt_BR.ISO8859-1`	`pt_BR.po`	ISO 8859-1
Russo	Rússia	`ru_RU.KOI8-R`	`ru_RU.po`	KOI8-R
Turco	Turquia	`tr_TR.ISO8859-9`	`tr_TR.po`	ISO 8859-9
Chinês	China	`zh_CN.UTF-8`	`zh_CN.po`	UTF-8
Chinês	Taiwan	`zh_TW.UTF-8`	`zh_TW.po`	UTF-8

As traduções estão em subdiretórios do diretório principal da documentação, aqui assumido como `~/doc/` como mostrado em Seção 1.1, “Quick Start”. Por exemplo, as traduções em alemão estão localizadas em `~/doc/de_DE.ISO8859-1/` e as traduções em francês estão em `~/doc/fr_FR.ISO8859-1/` .

Cada diretório de idiomas contém subdiretórios separados para os tipos de documentos, geralmente `articles/` e `books/`.

A combinação desses nomes de diretórios fornece o caminho completo para um artigo ou livro. Por exemplo, a tradução Francesa do artigo NanoBSD está em `~/doc/fr_FR.ISO8859-1/articles/nanobsd/`, e a tradução em Mongol do manual está em `~/doc/mn_MN.UTF-8/books/handbook/`.

Um novo diretório de idioma deve ser criado ao traduzir um documento para um novo idioma. Se o diretório de idiomas já existir, somente um subdiretório no diretório `articles/` ou `books/` será necessário.

As compilações da documentação do FreeBSD são controladas por um `Makefile` no mesmo diretório. Com artigos simples, o `Makefile` muitas vezes pode ser copiado literalmente do diretório original em inglês. O processo de tradução combina vários arquivos `book.xml` e `chapter.xml` de livros em um único arquivo, portanto, o `Makefile` para as traduções de livros deve ser copiado e modificado.

Exemplo 12.1. Criando uma Tradução em Espanhol do Porter's Handbook

Crie uma nova tradução em Espanhol do Porter's Handbook. O original é um livro em `~/doc/en_US.ISO8859-1/books/porters-handbook/`.

1. O diretório de livros em Espanhol `~/doc/es_ES.ISO8859-1/books/` já existe, portanto, apenas um novo subdiretório para o Porter's Handbook é necessário:

```
% cd ~/doc/es_ES.ISO8859-1/books/
% svn mkdir porters-handbook
A         porters-handbook
```

2. Copie o `Makefile` do livro original:

```
% cd ~/doc/es_ES.ISO8859-1/books/porters-handbook
% svn cp ~/doc/en_US.ISO8859-1/books/porters-handbook/Makefile .
A         Makefile
```

   Modifique o conteúdo do `Makefile` para esperar apenas um único `book.xml`:

```
#
# $FreeBSD$
#
# Build the FreeBSD Porter's Handbook.
#

MAINTAINER=doc@FreeBSD.org

DOC?= book

FORMATS?= html-split

INSTALL_COMPRESSED?= gz
INSTALL_ONLY_COMPRESSED?=

# XML content
SRCS=  book.xml

# Images from the cross-document image library
IMAGES_LIB+=    callouts/1.png
IMAGES_LIB+=    callouts/2.png
IMAGES_LIB+=    callouts/3.png
IMAGES_LIB+=    callouts/4.png
IMAGES_LIB+=    callouts/5.png
IMAGES_LIB+=    callouts/6.png
IMAGES_LIB+=    callouts/7.png
```

```
IMAGES_LIB+=    callouts/8.png
IMAGES_LIB+=    callouts/9.png
IMAGES_LIB+=    callouts/10.png
IMAGES_LIB+=    callouts/11.png
IMAGES_LIB+=    callouts/12.png
IMAGES_LIB+=    callouts/13.png
IMAGES_LIB+=    callouts/14.png
IMAGES_LIB+=    callouts/15.png
IMAGES_LIB+=    callouts/16.png
IMAGES_LIB+=    callouts/17.png
IMAGES_LIB+=    callouts/18.png
IMAGES_LIB+=    callouts/19.png
IMAGES_LIB+=    callouts/20.png
IMAGES_LIB+=    callouts/21.png

URL_RELPREFIX?= ../../../..
DOC_PREFIX?= ${.CURDIR}/../../..

.include "${DOC_PREFIX}/share/mk/doc.project.mk"
```

Agora a estrutura do documento está pronta para o tradutor começar a tradução com `make po`.

Exemplo 12.2. Criando uma tradução em Francês do Artigo Chaves Open PGP

Crie uma nova tradução em Francês do artigo Chaves Open PGP. O original é um artigo em `~/doc/en_US.ISO8859-1/articles/pgpkeys/` .

1. O diretório de artigos em Francês `~/doc/fr_FR.ISO8859-1/articles/` já existe, portanto, apenas um novo subdiretório para o artigo Chaves Open PGP é necessário:

```
% cd ~/doc/fr_FR.ISO8859-1/articles/
% svn mkdir pgpkeys
A         pgpkeys
```

2. Copie o `Makefile` do artigo original:

```
% cd ~/doc/fr_FR.ISO8859-1/articles/pgpkeys
% svn cp ~/doc/en_US.ISO8859-1/articles/pgpkeys/Makefile .
A         Makefile
```

Verifique o conteúdo do `Makefile`. Este é um artigo simples e neste caso, o `Makefile` pode ser usado sem altaração. A string `$FreeBSD...$` na segunda linha será substituída pelo sistema de controle de versão quando este arquivo sofrer um commit.

```
#
# $FreeBSD$
#
# Article: PGP Keys

DOC?= article

FORMATS?= html
WITH_ARTICLE_TOC?= YES

INSTALL_COMPRESSED?= gz
INSTALL_ONLY_COMPRESSED?=

SRCS=   article.xml
```

```
# To build with just key fingerprints, set FINGERPRINTS_ONLY.

URL_RELPREFIX?= ../../../..
DOC_PREFIX?=    ${.CURDIR}/../../..

.include "${DOC_PREFIX}/share/mk/doc.project.mk"
```

Com a estrutura do documento completa, o arquivo PO pode ser criado com `make po`.

12.4. Traduzindo

O sistema gettext reduz bastante o número de itens que devem ser rastreados por um tradutor. As strings a serem traduzidas são extraídas do documento original em um arquivo PO. Em seguida, um editor PO é usado para inserir as traduções de cada string.

O sistema de tradução PO do FreeBSD não sobrescreve os arquivos PO, portanto a etapa de extração pode ser executada a qualquer momento para atualizar o arquivo PO.

Um editor PO é usado para editar o arquivo. editores/poedit é usado nestes exemplos porque é simples e tem requisitos mínimos. Outros editores PO oferecem recursos para facilitar o trabalho de tradução. A Coleção de Ports oferece vários desses editores, incluindo o devel/gtranslator.

É importante preservar o arquivo PO. Ele contém todo o trabalho que os tradutores fizeram.

Exemplo 12.3. Traduzindo o Porter's Handbook para o Espanhol

Digite as traduções para o Espanhol do conteúdo do Porter's Handbook.

1. Mude para o diretório Espanhol do Porter's Handbook e atualize o arquivo PO. O arquivo PO gerado é chamado `es_ES.po` como mostrado em Tabela 12.1, “Nomes de Idioma”.

   ```
   % cd ~/doc/es_ES.ISO8859-1/books/porters-handbook
   % make po
   ```

2. Realize as traduções usando um editor de PO:

   ```
   % poedit es_ES.po
   ```

12.5. Dicas para Tradutores

12.5.1. Preservando Tags XML

Preserve as tags XML mostradas no original em Inglês.

Exemplo 12.4. Preservando Tags XML

Inglês original:

```
If <acronym> NTP</acronym>  is not being used
```

Tradução para o Espanhol:

```
Si <acronym> NTP</acronym>  no se utiliza
```

12.5.2. Preservando Espaços

Preserve os espaços existentes no início e no final das strings a serem traduzidas. A versão traduzida também deve ter esses espaços.

12.5.3. Tags

O conteúdo de algumas tags devem ser copiadas igualmente, sem realizar tradução:

- `<citerefentry>`
- `<command>`
- `<filename>`
- `<literal>`
- `<manvolnum>`
- `<orgname>`
- `<package>`
- `<programlisting>`
- `<prompt>`
- `<refentrytitle>`
- `<screen>`
- `<userinput>`
- `<varname>`

12.5.4. Strings `$FreeBSD$`

As strings da versão $FreeBSD$ usadas nos arquivos requerem tratamento especial. Nos exemplos como Exemplo 12.1, “Criando uma Tradução em Espanhol do Porter's Handbook”, essas strings não devem ser expandidas. Os documentos em inglês usam entidades `&dollar;` para evitar a inclusão de sinais reais de dólar no arquivo:

```
&dollar;FreeBSD&dollar;
```

As entidades `&dollar;` não são vistas como cifrões pelo sistema de controle de versão e, portanto, a string não é expandida em uma string de versão.

Quando um arquivo PO é criado, as entidades `&dollar;` usadas nos exemplos são substituídas por cifrões reais. A sequência literal `$FreeBSD$` será expandida incorretamente pelo sistema de controle de versão quando o arquivo sofrer algum commit.

A mesma técnica usada nos documentos em Inglês pode ser usada na tradução. O `&dollar;` é usado para substituir o sinal de dólar na tradução inserida no editor PO:

```
&dollar;FreeBSD&dollar;
```

12.6. Compilando um Documento Traduzido

Uma versão traduzida do documento original pode ser criada a qualquer momento. Quaisquer porções não traduzidas do original serão incluídas em Inglês no documento resultante. A maioria dos editores PO tem um indicador que mostra quanto da tradução foi realizada. Isso torna mais fácil para o tradutor ver quantas strings foram traduzidas para tornar a compilação do documento final utiilizável.

Exemplo 12.5. Construindo o Porter's Handbook Espanhol

Compile e visualize a versão em Espanhol do Porter's Handbook criado em um exemplo anterior.

1. Compile o documento traduzido. Como o original é um livro, o documento gerado é `book.xml` .

```
% cd ~/doc/es_ES.ISO8859-1/books/porters-handbook
% make tran
```

2. Renderize o `book.xml` traduzido para HTML e visualize-o com o Firefox. Este é o mesmo procedimento usado com a versão em Inglês dos documentos, e outros `FORMATS` podem ser usados aqui da mesma maneira. Veja Tabela 5.1, “Formatos de Saída Comuns”.

```
% make FORMATS=html
% firefox book.html
```

12.7. Submetendo a Nova Tradução

Prepare os novos arquivos de tradução para envio. Isso inclui adicionar os arquivos ao sistema de controle de versão, definir propriedades adicionais e criar um diff para a submissão.

Os arquivos diff criados por esses exemplos podem ser anexados a um relatório de bug de documentação ou revisão de código.

Exemplo 12.6. Tradução Espanhola do Artigo NanoBSD

1. Adicione a string FreeBSD na primeira linha do arquivo PO:

```
#$FreeBSD$
```

2. Adicione o `Makefile` , o arquivo PO e o arquivo traduzido XML que foi gerado para o controle de versão:

```
% cd ~/doc/es_ES.ISO8859-1/articles/nanobsd/
% ls
Makefile article.xml es_ES.po
% svn add Makefile article.xml es_ES.po
A         Makefile
A         article.xml
A         es_ES.po
```

3. Configure as propriedades Subversion `svn:keywords` nesses arquivos para `FreeBSD=%H` para que as strings `$FreeBSD$` sejam expandidas com o caminho, revisão, data e autor quando o arquivo sofrer o commit:

```
% svn propset svn:keywords FreeBSD=%H Makefile article.xml es_ES.po
property 'svn:keywords' set on 'Makefile'
```

```
property 'svn:keywords' set on 'article.xml'
property 'svn:keywords' set on 'es_ES.po'
```

4. Defina os tipos MIME dos arquivos. Estes são `text/xml` para livros e artigos, e `text/x-gettext-translation` para o arquivo PO.

```
% svn propset svn:mime-type text/x-gettext-translation es_ES.po
property 'svn:mime-type' set on 'es_ES.po'
% svn propset svn:mime-type text/xml article.xml
property 'svn:mime-type' set on 'article.xml'
```

5. Crie um diff dos novos arquivos a partir do diretório base `~/doc/` para que o caminho completo seja mostrado com os nomes dos arquivos. Isso ajuda os committers a identificar o diretório do idioma de destino.

```
% cd ~/doc
svn diff es_ES.ISO8859-1/articles/nanobsd/ > /tmp/es_nanobsd.diff
```

Exemplo 12.7. Tradução Coreana UTF-8 do Artigo Explicando o BSD

1. Adicione a string FreeBSD na primeira linha do arquivo PO:

```
#$FreeBSD$
```

2. Adicione o `Makefile` , o arquivo PO e o arquivo traduzido XML que foi gerado para o controle de versão:

```
% cd ~/doc/ko_KR.UTF-8/articles/explaining-bsd/
% ls
Makefile article.xml ko_KR.po
% svn add Makefile article.xml ko_KR.po
A         Makefile
A         article.xml
A         ko_KR.po
```

3. Configure as propriedades Subversion `svn:keywords` nesses arquivos para `FreeBSD=%H` para que as strings `$FreeBSD$` sejam expandidas com o caminho, revisão, data e autor quando o arquivo sofrer o commit:

```
% svn propset svn:keywords FreeBSD=%H Makefile article.xml ko_KR.po
property 'svn:keywords' set on 'Makefile'
property 'svn:keywords' set on 'article.xml'
property 'svn:keywords' set on 'ko_KR.po'
```

4. Defina os tipos MIME dos arquivos. Como esses arquivos usam o conjunto de caracteres UTF-8, isso também é especificado. Para evitar que o sistema de controle de versão confunda esses arquivos com arquivos binários, a propriedade `fbsd:notbinary` também é configurada:

```
% svn propset svn:mime-type 'text/x-gettext-translation; charset=UTF-8' ko_KR.po
property 'svn:mime-type' set on 'ko_KR.po'
% svn propset fbsd:notbinary yes ko_KR.po
property 'fbsd:notbinary' set on 'ko_KR.po'
% svn propset svn:mime-type 'text/xml; charset=UTF-8' article.xml
property 'svn:mime-type' set on 'article.xml'
% svn propset fbsd:notbinary yes article.xml
property 'fbsd:notbinary' set on 'article.xml'
```

5. Crie um diff desses novos arquivos no diretório base `~/doc/` :

```
% cd ~/doc
```

```
svn diff ko_KR.UTF-8/articles/explaining-bsd > /tmp/ko-explaining.diff
```

Capítulo 13. Páginas de Manual

13.1. Introdução

Páginas de manual, geralmente abreviadas como *man pages*, foram concebidas como lembretes prontamente disponíveis para sintaxe de comandos, detalhes de drivers de dispositivos ou formatos de arquivos de configuração. Elas se tornaram uma referência rápida extremamente valiosa de linha de comando para usuários, administradores de sistema e programadores.

Embora tenham sido planejados como material de referência em vez de tutoriais, as seções EXEMPLOS das páginas de manual geralmente fornecem casos de uso detalhados.

Páginas de manual são geralmente mostradas interativamente pelo comando man(1). Quando o usuário digita `man ls`, uma pesquisa é executada para uma página de manual que corresponde a `ls`. O primeiro resultado correspondente é exibido.

13.2. Seções

As páginas de manual são agrupadas em *seções*. Cada seção contém páginas de manual para uma categoria específica de documentação:

Número da Seção	Categoria
1	Comandos Gerais
2	System Calls
3	Library Functions
4	Interfaces de Kernel
5	Formatos de Arquivo
6	Jogos
7	Diversos
8	System Manager
9	Desenvolvedor Kernel

13.3. Marcação

Vários formulários de marcação e programas de renderização foram usados para páginas de manual. O FreeBSD usou groff(7) e o mais recente mandoc(1). A maioria das páginas de manual do FreeBSD, e todas as novas, usam o formulário mdoc(7) de marcação. Esta é uma marcação simples baseada em linhas que é razoavelmente expressiva. É principalmente semântico: partes do texto são marcadas para o que são, em vez de como devem aparecer quando renderizadas. Existe alguma marcação baseada em aparência que geralmente é melhor evitar.

O código fonte de página de manual geralmente é interpretada e exibido na tela interativamente. Os arquivos fontes podem ser arquivos de texto comuns ou compactados com gzip(1) para economizar espaço.

As páginas de manual também podem ser renderizadas para outros formatos, incluindo PostScript para impressão ou geração de PDF. Veja man(1).

Dica

O teste de uma nova página de manual pode ser um desafio quando o arquivo não está localizado no caminho de pesquisa normal da páginas de manual. man(1) também não procura no diretório atual. Se a nova página de manual estiver no diretório atual, prefixe o nome do arquivo com um `./`:

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

Um caminho absoluto também pode ser usado:

```
% man /home/xsmith/mynewmanpage.8
```

13.3.1. Seções de Página de Manual

Páginas de manual são compostas por várias seções padrão. Cada seção tem um título em letras maiúsculas e as seções de um determinado tipo de página de manual aparecem em uma ordem específica. Para uma página de manual do Comando Geral da categoria 1, as seções são:

Nome da Seção	Descrição
NAME	Nome do Comando
SYNOPSIS	Formato das opções e argumentos
DESCRIPTION	Descrição da finalidade e uso
ENVIRONMENT	Configurações de ambiente que afetam a operação
EXIT STATUS	Códigos de erro retornados na saída
EXAMPLES	Exemplos de uso
COMPATIBILITY	Compatibilidade com outras implementações
SEE ALSO	Referência cruzada para páginas de manual relacionadas
STANDARDS	Compatibilidade com padrões como o POSIX
HISTORY	História de implementação
BUGS	Bugs conhecidos
AUTHORS	Pessoas que criaram o comando ou escreveram a página de manual.

Algumas seções são opcionais e a combinação de seções para um tipo específico de página manual pode variar. Exemplos dos tipos mais comuns são mostrados mais adiante neste capítulo.

13.3.2. Macros

A marcação mdoc(7) é baseada em *macros*. As linhas que começam com um ponto contêm comandos de macro, com duas ou três letras. Por exemplo, veja esta parte da página de manual do ls(1):

```
.Dd December 1, 2015  ❶
.Dt LS 1
.Sh NAME  ❷
.Nm ls
.Nd list directory contents
.Sh SYNOPSIS  ❸
.Nm  ❹
```

```
.Op Fl -libxo  ❺
.Op Fl ABCFGHILPRSTUWZabcdfghiklmnopqrstuwxy1,  ❻
.Op Fl D Ar format  ❼
.Op Ar  ❽
.Sh DESCRIPTION  ❾
For each operand that names a
.Ar file
of a type other than
directory,
.Nm
displays its name as well as any requested,
associated information.
For each operand that names a
.Ar file
of type directory,
.Nm
displays the names of files contained
within that directory, as well as any requested, associated
information.
```

❶ O *Document date* e *Document title* são definidos.
❷ O *Cabeçalho da Seção* para a seção NAME é definido. Em seguida, são definidos o *Nome* do comando e uma *Descrição do Nome* de uma linha.
❸ A seção SYNOPSIS começa. Esta seção descreve as opções e argumentos de linha de comando que são aceitos.
❹ *Nome* (`.Nm`) já foi definido, e repeti-lo aqui apenas exibe o valor definido no texto.
❺ Uma *Optional Flag* chamada `-libxo` é mostrada. A macro `Fl` adiciona um traço ao início das flags, portanto, isso aparece na página de manual como `--libxo`.
❻ Uma longa lista de sinalizadores opcionais de caracteres únicos é apresentada.
❼ Uma flag opcional `-D` é definida. Se a flag `-D` for informada, ela deve ser seguida por um *Argumento*. O argumento é um *format*, uma string que diz ls(1) o que exibir e como exibi-lo. Detalhes sobre a string de formato são fornecidos posteriormente na página de manual.
❽ Um argumento opcional final é definido. Como nenhum nome é especificado para o argumento, o padrão `file ...` é usado.
❾ O *Cabeçalho da Seção* para a seção DESCRIPTION é definido.

Quando renderizado com o comando `man ls`, o resultado exibido na tela é semelhante ao seguinte:

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

NAME
     ls — list directory contents

SYNOPSIS
     ls [--libxo] [-ABCFGHILPRSTUWZabcdfghiklmnopqrstuwxy1,] [-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.
```

Valores opcionais são mostrados entre colchetes.

13.3.3. Diretrizes de Marcação

A linguagem de marcação mdoc(7) não é muito rigorosa. Para maior clareza e consistência, o projeto Documentação do FreeBSD adiciona algumas diretrizes de estilo adicionais:

Apenas a primeira letra das macros é maiúscula
: Sempre use maiúsculas para a primeira letra de uma macro e minúscula para as letras restantes.

Comece novas frases em novas linhas
: Inicie uma nova frase em uma nova linha, não a inicie na mesma linha de uma frase existente.

Atualizar `.Dd` ao fazer alterações não triviais em uma página de manual
: A *Data do documento* informa o leitor sobre a última vez que a página de manual foi atualizada. É importante atualizar sempre que alterações não triviais forem feitas nas páginas de manual. Alterações triviais, como correções ortográficas ou de pontuação que não afetam o uso, podem ser feitas sem atualizar `.Dd`.

Apresentando exemplos
: Apresente exemplos ao leitor sempre que possível. Mesmo exemplos triviais são valiosos, porque o que é trivial para o escritor não é necessariamente trivial para o leitor. Três exemplos são um bom objetivo. Um exemplo trivial mostra os requisitos mínimos, um exemplo afundo mostra o uso real e um exemplo detalhado demonstra uma funcionalidade incomum ou não óbvia.

Inclua a licença BSD
: Inclua a licença BSD em novas páginas de manual. A licença preferencial está disponível no Guia dos Committer's.

13.3.4. Truques de Marcação

Adicione um espaço antes da pontuação em uma linha com macros. Exemplo:

```
.Sh SEE ALSO
.Xr geom 4 ,
.Xr boot0cfg 8 ,
.Xr geom 8 ,
.Xr gptboot 8
```

Observe como as vírgulas no final das linhas `.Xr` foram colocadas após um espaço. A macro `.Xr` espera dois parâmetros, o nome de uma página de manual externa e um número de seção. O espaço separa a pontuação do número da seção. Sem o espaço, os links externos apontariam incorretamente para a seção `4,` ou `8,`.

13.3.5. Macros Importantes

Algumas macros muito comuns serão mostradas aqui. Para obter mais exemplos de uso, consulte mdoc(7), groff_mdoc(7), ou procure por uso real no diretório `/usr/share/man/man*` . Por exemplo, para procurar exemplos da macro `.Bd` *Begin display*:

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

13.3.5.1. Macros Organizacionais

Algumas macros são usadas para definir blocos lógicos de uma página de manual.

Macro Organizacional	Uso
`.Sh`	Section header (Cabeçalho da seção). Seguido pelo nome da seção, tradicionalmente com os caracteres todos em maiúsculas. Pense neles como títulos de capítulos.
`.Ss`	Subsection header (Cabeçalho da subseção). Seguido pelo nome da subseção. Usado para dividir uma seção `.Sh` em subseções.
`.Bl`	Begin list. Comece uma lista de itens.
`.El`	End a list (Finalize uma lista).
`.Bd`	Begin display (Comece a exibição). Comece em uma área especial de texto, como uma área recuada.
`.Ed`	End display (Termine a exibição).

13.3.5.2. Macros Inline

Muitas macros são usadas para marcar texto embutido.

Macro Inline	Uso
`.Nm`	Name. Chamado com um nome como parâmetro no primeiro uso, usado depois sem o parâmetro para exibir o nome que já foi definido.
`.Pa`	Path to a file (Caminho para um arquivo). Usado para marcar nomes de arquivos e caminhos de diretório.

13.4. Exemplo de Estruturas de Página de Manual

Esta seção mostra o conteúdo mínimo desejável para um página de manual para várias categorias comuns de páginas de manual.

13.4.1. Seção 1 ou 8 sobre um comando

A estrutura básica preferida para uma seção 1 ou 8 sobre um comando:

```
.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
```

13.4.2. Seção 4 sobre um Driver de Dispositivo

A estrutura básica preferida para a seção 4 sobre um driver de dispositivo:

```
.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"
.Ed
.Pp
To load the driver as a module at boot, add this line to
.Xr loader.conf 5 :
.Bd -literal -offset indent
exampledriver_load="YES"
.Ed
```

```
.Sh DESCRIPTION
The
.Nm
driver provides an opportunity to show a skeleton or template
file for section 4 manual pages.
.Sh HARDWARE
The
.Nm
driver supports these cards from the aptly-named Nonexistent
Technologies:
.Pp
.Bl -bullet -compact
.It
NT X149.2 (single and dual port)
.It
NT X149.8 (single port)
.El
.Sh DIAGNOSTICS
.Bl -diag
.It "flashing green light"
Something bad happened.
.It "flashing red light"
Something really bad happened.
.It "solid black light"
Power cord is unplugged.
.El
.Sh SEE ALSO
.Xr example 8
.Sh HISTORY
The
.Nm
device driver first appeared in
.Fx 49.2 .
.Sh AUTHORS
.An Firstname Lastname Aq Mt flastname@example.com
```

13.4.3. Seção 5 sobre um Arquivo de Configuração

A estrutura básica preferida para a seção 5 sobre um arquivo de configuração:

```
.Dd August 25, 2017
.Dt EXAMPLECONF 5
.Os
.Sh NAME
.Nm example.conf
.Nd "config file to demonstrate section 5 man pages"
.Sh DESCRIPTION
.Nm
is an example configuration file.
.Sh SEE ALSO
.Xr example 8
.Sh AUTHORS
.An Firstname Lastname Aq Mt flastname@example.com
```

13.5. Exemplos de páginas de manuais para usar como modelos

Algumas destas páginas de manual são adequadas para serem usadas como exemplos detalhados.

Página Manual	Caminho para o local de origem
cp(1)	`/usr/src/bin/cp/cp.1`
vt(4)	`/usr/src/share/man/man4/vt.4`
crontab(5)	`/usr/src/usr.sbin/cron/crontab/crontab.5`

Página Manual	Caminho para o local de origem
gpart(8)	`/usr/src/sbin/geom/class/part/gpart.8`

13.6. Recursos

Recursos para escritores de páginas manuais:

- man(1)
- mandoc(1)
- groff_mdoc(7)
- Manuais Práticos do UNIX: mdoc
- História das Man pages do UNIX

Capítulo 14. Estilo de escrita

14.1. Dicas

A documentação técnica pode ser melhorada pelo uso consistente de vários princípios. A maioria destes pode ser classificada em três objetivos: *ser claro*, *ser completo* e *ser conciso*. Essas metas podem entrar em conflito umas com as outras. Uma boa escrita consiste em um equilíbrio entre eles.

14.1.1. Seja claro

A clareza é extremamente importante. O leitor pode ser um novato ou ler o documento em um segundo idioma. Esforce-se por um texto simples e descomplicado que explique claramente os conceitos.

Evite discurso florido ou embelezado, piadas ou expressões coloquiais. Escreva da maneira mais simples e clara possível. Um texto simples é mais fácil de se entender e de se traduzir.

Mantenha as explicações o mais curtas, simples e claras possível. Evite frases vazias como "a fim de" as quais normalmente significam apenas um "para". Evite palavras potencialmente paternalistas tais como "basicamente". Evite termos latinos como "i.e." ou "cf.", os quais podem ser desconhecidos fora de grupos acadêmicos ou científicos.

Escreva em um estilo formal. Evite dirigir-se ao leitor como "você". Por exemplo, digamos "copie o arquivo para `/tmp`" em vez de "você pode copiar o arquivo para `/tmp`".

Dê exemplos claros, corretos, e *testados*. Um exemplo trivial é melhor do que nenhum exemplo. Um bom exemplo é ainda melhor. Não dê exemplos ruins, identificáveis por desculpas ou frases como "mas realmente isso nunca deve ser feito dessa forma". Exemplos ruins são piores que nenhum exemplo. Dê bons exemplos, porque *mesmo quando avisado para não usar o exemplo como mostrado* , o leitor normalmente só usa o exemplo como mostrado.

Evite *palavras vazias* como "deveria", "poderia", "tentaria", ou "podia". Estas palavras implicam que o autor não tem certeza dos fatos e cria dúvidas no leitor.

Da mesma forma, dê instruções como comandos imperativos: não utilize "você deve fazer isso", mas apenas "faça isso".

14.1.2. Seja completo

Não faça suposições sobre as habilidades do leitor. Diga-lhes o que precisam saber. Dê links para outros documentos para fornecer informações básicas sem precisar recriá-las. Coloque-se no lugar do leitor, antecipe as perguntas que eles farão e responda-os.

14.1.3. Seja conciso

Embora as funcionalidades devam ser documentadas completamente, às vezes existe tanta informação que o leitor não consegue encontrar facilmente os detalhes específicos de que necessita. O equilíbrio entre ser completo e ser conciso é um desafio. Uma abordagem é ter uma introdução e, em seguida, uma seção de "início rápido" que descreve a situação mais comum, seguida por uma seção de referência aprofundada.

14.2. Diretrizes

Para promover a consistência entre os inúmeros autores da documentação do FreeBSD, algumas diretrizes foram elaboradas para os autores seguirem.

Use a Ortografia do Inglês Americano
: Existem várias variantes do inglês, com grafias diferentes para a mesma palavra. Onde as grafias diferem, use a variante do inglês americano. “color”, não “colour”, “rationalize”, não “rationalise”, e assim por diante.

> **Nota**
>
> O uso do inglês britânico pode ser aceito no caso de um artigo contribuído, no entanto, a ortografia deve ser consistente em todo o documento. Os outros documentos, como livros, site, páginas de manual, etc., terão que usar o inglês americano.

Não use contrações
: Não use contrações. Sempre soletre a frase na íntegra. “Do not” é a forma correta, “Don't” é a errada.

  Evitar contrações contribui para um tom mais formal, é mais preciso e é um pouco mais fácil para os tradutores.

Use a vírgula serial
: Em uma lista de itens dentro de um parágrafo, separe cada item dos outros com uma vírgula. Separe o último item dos outros com uma vírgula e a letra “e”.

  Por exemplo:

  > Esta é uma lista de um, dois e três itens.

  É uma lista de três itens, “um”, “dois” e “três”, ou uma lista de dois itens, “um” e “dois e três”?

  É melhor ser explícito e incluir uma vírgula serial:

  > Esta é uma lista de um, dois, e três itens.

Evite frases redundantes
: Não use frases redundantes. Em particular, “o comando”, “o arquivo” e o “comando man” são frequentemente redundantes.

  Por exemplo, comandos:

  Errado: Use o comando `svn` para atualizar o código fonte.

  Correto: Use o `svn` para atualizar o código fonte.

  Nomes de arquivo:

  Errado:... no nome do arquivo `/etc/rc.local` ...

  Correto:... no `/etc/rc.local` ...

  Referências de páginas de manual (o segundo exemplo usa `citerefentry` com a entidade `&man.csh.1;`):.

  Errado: Veja `man csh` para mais informações.

  Correto: Veja csh(1).

Dois espaços entre frases
: Sempre use dois espaços entre as sentenças, pois isso melhora a legibilidade e facilita o uso de ferramentas como o Emacs.

  Um período e espaços seguidos por uma letra maiúscula nem sempre marcam uma nova sentença, especialmente em nomes. “Jordan K. Hubbard” é um bom exemplo. Ele tem um capital `H` após um período e um espaço, e certamente não é uma nova sentença.

Para mais informações sobre o estilo de escrita, consulte Elementos de Estilo, de William Strunk.

14.3. Guia de estilo

Para manter o código fonte da documentação consistente quando muitas pessoas diferentes a estiverem editando, siga estas convenções de estilo.

14.3.1. Letter Case

As tags são digitadas em minúsculas, `para`, *não* `PARA`.

O texto que aparece em contextos SGML é geralmente escrito em letras maiúsculas, `<! ENTITY ... >`, e `<! DOCTYPE… >`, *não* `<! entity… >` e `<! doctype… >`.

14.3.2. Siglas

Os acrônimos devem ser definidos na primeira vez que aparecerem em um documento, como em: “Network Time Protocol (NTP)”. Depois que o acrônimo tiver sido definido, use apenas o acrônimo, a menos que faça mais sentido contextualmente usar todo o termo. Acrônimos geralmente são definidos apenas uma vez por capítulo ou por documento.

Todos os acrônimos devem ser colocados em tags `acronym`.

14.3.3. Indentação

A primeira linha em cada arquivo começa sem recuo, *independentemente* do nível de recuo do arquivo que pode conter o arquivo atual.

Tags de abertura aumentam o nível de recuo por dois espaços. Tags de fechamento diminuem o nível de recuo por dois espaços. Blocos de oito espaços no início de uma linha devem ser substituídos por um tab. Não use espaços na frente das tabs e não adicione espaço em branco extra no final de uma linha. O conteúdo nos elementos deve ser indentado por dois espaços se o conteúdo for executado em mais de uma linha.

Por exemplo, a fonte desta seção é assim:

```
<chapter>
  <title>...</title>

  <sect1>
    <title>...</title>

    <sect2>
      <title>Indentation</title>

      <para>The first line in each file starts with no indentation,
 <emphasis>regardless</emphasis>  of the indentation level of
 the file which might contain the current file.</para>

      ...
    </sect2>
  </sect1>
</chapter>
```

Tags contendo atributos longos seguem as mesmas regras. Seguir as regras de recuo neste caso ajuda editores e escritores a ver qual conteúdo está dentro das tags:

```
<para>See the <link
    linkend="gmirror-troubleshooting">  Troubleshooting</link>
  section if there are problems booting.  Powering down and
  disconnecting the original <filename>ada0</filename>  disk
```

```
  will allow it to be kept as an offline backup.</para>

<para>It is also possible to journal the boot disk of a &os;
  system.  Refer to the article <link
    xlink:href="&url.articles.gjournal-desktop;">  Implementing UFS
    Journaling on a Desktop PC</link>  for detailed
  instructions.</para>
```

Quando um elemento é muito longo para caber no restante de uma linha sem quebrá-la, mover a tag inicial para a próxima linha pode facilitar a leitura do código. Neste exemplo, o elemento `systemitem` foi movido para a próxima linha para evitar a quebra e o recuo:

```
<para>With file flags, even
  <systemitem class="username">  root</systemitem>  can be
  prevented from removing or altering files.</para>
```

Configurações para ajudar vários editores de texto a operar em conformidade com estas diretrizes podem ser encontradas em Capítulo 15, *Configuração do Editor*.

14.3.4. Estilo de Tag

14.3.4.1. Espaçamento de tag

Tags que começam no mesmo recuo como uma tag anterior devem ser separadas por uma linha em branco, e aquelas que não estão no mesmo recuo como uma tag anterior não devem:

```
<article lang='en'>
  <articleinfo>
    <title>NIS</title>

    <pubdate>October 1999</pubdate>

    <abstract>
      <para>...
 ...
 ...</para>
    </abstract>
  </articleinfo>

  <sect1>
    <title>...</title>

    <para>...</para>
  </sect1>

  <sect1>
    <title>...</title>

    <para>...</para>
  </sect1>
</article>
```

14.3.4.2. Separando Tags

Tags como `itemizedlist` , que sempre terão outras tags dentro delas, e de fato não pegam dados de caractere, estão sempre sozinhas em uma linha.

Tags como `para` e `term` não precisam de outras tags para conter dados de caracteres normais, e seus conteúdos começam imediatamente após a tag, *na mesma linha* .

O mesmo se aplica quando esses dois tipos de tags fecham.

Isso leva a um problema óbvio ao misturar essas tags.

Quando uma tag inicial que não pode conter dados de caractere segue diretamente uma tag do tipo que requer outras tags dentro dela para usar dados de caractere, elas estão em linhas separadas. A segunda tag deve estar corretamente recuada.

Quando uma tag que pode conter dados de caractere é fechada diretamente após uma tag que não pode conter dados de caractere fechados, eles coexistem na mesma linha.

14.3.5. Mudanças no espaço em branco

Não faça commit de mudanças no conteúdo ao mesmo tempo em que faz mudanças na formatação.

Quando as alterações de conteúdo e espaço em branco são mantidas separadas, as equipes de tradução podem ver facilmente se uma alteração foi de um conteúdo que deve ser traduzido ou apenas espaço em branco.

Por exemplo, se duas sentenças foram adicionadas a um parágrafo para que os comprimentos de linha passem de 80 colunas, primeiro faça commit da alteração com as linhas longas demais. Em seguida, corrija a quebra de linha e confirme essa segunda alteração. Na mensagem de confirmação da segunda alteração, indique que esta é uma alteração somente de espaço em branco a qual pode ser ignorada pelos tradutores.

14.3.6. Espaços Não Separáveis (Non-Breaking Space)

Evite quebras de linha em locais onde elas pareçam feias ou dificultem o entendimento de uma frase. Quebras de linha dependem da largura do meio de saída escolhido. Em particular, a visualização da documentação em HTML com um navegador de texto pode levar a parágrafos mal formatados como o seguinte:

```
A capacidade de dados varia de 40 MB a 15
GB Compressão de hardware…
```

A entidade geral ` ` proíbe as quebras de linha entre as partes que estão juntas. Use espaços não separáveis nos seguintes locais:

- entre números e unidades:

  ```
  57600 bps
  ```

- entre nomes de programas e números de versão:

  ```
  &os; 9.2
  ```

- entre nomes com várias palavras (use com cautela ao aplicar isso em nomes de mais de 3-4 palavras como “The FreeBSD Portuguese Portuguese Documentation Project”):

  ```
  Sun &Microsystems
  ```

14.4. Lista de palavras

Esta lista de palavras mostra a ortografia e letras maiúsculas corretas quando usadas na documentação do FreeBSD. Se uma palavra não estiver nesta lista, pergunte sobre isso na lista de discussão do projeto de documentação do FreeBSD.

Palavra	Código XML	Notas
CD-ROM	`<acronym> CD-ROM </acronym>`	
DoS (negação de serviço)	`<acronym> DoS</acronym>`	
email		
sistema de arquivo		
IPsec		

Palavra	Código XML	Notas
Internet		
página de manual		
servidor de email		
nome do servidor		
Coleção de Ports		
somente leitura		
Soft Updates		
stdin	<varname> stdin</varname>	
stdout	<varname> stdout</varname>	
stderr	<varname> stderr</varname>	
Subversion	<application> Subversion </application>	Não se refira ao aplicativo Subversion como SVN em letras maiúsculas. Para se referir ao comando, use <command> svn</command> .
UNIX®	&unix;	
userland		coisas que se aplicam ao espaço do usuário, não ao kernel
servidor web		

Capítulo 15. Configuração do Editor

Ajustar a configuração do editor de texto pode tornar o trabalho nos arquivos da documentação mais rápido e fácil, além de ajudar os documentos a ficarem em conformidade com as diretrizes do FDP.

15.1. Vim

Instale-o a partir de editors/vim, editors/vim-console, ou editors/vim-tiny e siga as instruções de configuração em Seção 15.1.2, “Configuração”.

15.1.1. Uso

Pressione P para reformatar parágrafos ou texto selecionado no modo Visual. Pressione T para substituir grupos de oito espaços por um tab.

15.1.2. Configuração

Edite o `~/.vimrc` , adicionando estas linhas ao final do arquivo:

```
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/<CR>
    call ShowSpecial()
    call Set_Highlights()
    return 0
endfunction " Set_SGML()
```

15.2. Emacs

Instale-o a partir de editors/emacs ou editors/emacs-devel.

15.2.1. Validação

O modo nxml do Emacs usa esquemas NG relax compacto para validar o XML. Um esquema NG relax compacto para a extensão do FreeBSD para DocBook 5.0 está incluído no repositório de documentação. Para configurar o modo nxml para validar usando este esquema, crie `~/.emacs.d/schema/schemas.xml` e adicione estas linhas ao arquivo:

```
<locatingRules xmlns="http://thaiopensource.com/ns/locating-rules/1.0">
  <documentElement localName="section" typeId="DocBook">
  <documentElement localName="chapter" typeId="DocBook">
  <documentElement localName="article" typeId="DocBook">
  <documentElement localName="book" typeId="DocBook">
  <typeId id="DocBook" uri="/usr/local/share/xml/docbook/5.0/rng/docbook.rnc">
</locatingRules>
```

15.2.2. Revisão Automatizada com Flycheck e Igor

O pacote Flycheck está disponível no Emacs Lisp Package Archive da Milkypostman (MELPA). Se a MELPA ainda não estiver nos repositórios de pacotes do Emacs, ele pode ser adicionado executando

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

Adicione a linha ao arquivo de inicialização do Emacs (qualquer um deles, `~/.emacs` , `~/.emacs.el` , ou `~.emacs.d/init.el`) para tornar esta alteração permanente.

Para instalar o Flycheck, execute

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

Crie um verificador Flycheck para textproc/igor executando

```
(flycheck-define-checker igor
  "FreeBSD Documentation Project sanity checker.

See URLs https://www.freebsd.org/docproj/ and
http://www.freshports.org/textproc/igor/."
  :command ("igor" "-X" source-inplace)
  :error-parser flycheck-parse-checkstyle
  :modes (nxml-mode)
  :standard-input t)

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

Novamente, adicione essas linhas ao arquivo de inicialização do Emacs para tornar as mudanças permanentes.

15.2.3. Configurações Específicas da Documentação do FreeBSD

Para aplicar configurações específicas para o projeto de documentação do FreeBSD, crie o arquivo `.dir-locals.el` no diretório raiz do repositório de documentação e adicione estas linhas ao arquivo:

```
;;; Directory Local Variables
;;; For more information see (info "(emacs) Directory Variables")

((nxml-mode
  (eval . (turn-on-auto-fill))
  (fill-column . 70)
  (eval . (require 'flycheck))
  (eval . (flycheck-mode 1))
  (flycheck-checker . igor)
  (eval . (add-to-list 'rng-schema-locating-files "~/.emacs.d/schema/schemas.xml"))))
```

15.3. nano

Instale o aplicativo partir de editors/nano ou editors/nano-devel.

15.3.1. Configuração

Copie o arquivo com a amostra da regra para realce da sintaxe XML para o diretório inicial do usuário:

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

Use um editor para substituir as linhas do `~/.nanorc` referentes ao bloco `syntax "xml"` por estas regras:

```
syntax "xml" "\.([jrs]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
color ,red "^(([ ]{2})+|(TAB+))*[ ]{1}[^ ]{1}"
# lines longer than 70 characters
color ,yellow "^(.{71})|(TAB.{63})|(TAB{2}.{55})|(TAB{3}.{47}).+$"
```

Processe o arquivo para criar guias incorporadas:

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

15.3.2. Uso

Especifique opções úteis adicionais ao executar o editor:

```
% nano -AKipwz -r 70 -T8  chapter.xml
```

Usuários do csh(1) podem definir um alias em `~/.cshrc` para automatizar estas opções:

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

Depois que o alias é definido, as opções serão adicionadas automaticamente:

```
% nano chapter.xml
```

Capítulo 16. Veja também

Este documento não é deliberadamente uma discussão exaustiva de XML, das DTDs listadas e o Projeto de Documentação do FreeBSD. Para mais informações sobre estes, você é encorajado a consultar os seguintes sites.

16.1. O projeto de documentação do FreeBSD

- As Páginas Web do Projeto de Documentação do FreeBSD
- O Handbook do FreeBSD

16.2. XML

- Página W3C's XML Página Web SGML/XML

16.3. HTML

- O Consórcio World Wide Web
- A especificação HTML 4.0

16.4. DocBook

- O Comitê Técnico do DocBook , mantenedores do DocBook DTD
- DocBook: O Guia Definitivo , a documentação online do DocBook DTD
- O DocBook Open Repository contém folhas de estilo DSSSL e outros recursos para pessoas que usam DocBook

Apêndice A. Exemplos

Estes exemplos não são exaustivos - eles não contêm todos os elementos que podem ser desejáveis de usar, particularmente em relação ao inicio dos documentos (Front Matter). Para mais exemplos de marcação DocBook, examine o código XML para este e outros documentos disponíveis no repositório Subversion `doc` ou disponível on-line a partir de `http://svnweb.FreeBSD.org/doc/` .

A.1. `Livro` DocBook

Exemplo A.1. `Livro` DocBook

```
<!DOCTYPE book PUBLIC "-//FreeBSD//DTD DocBook XML V5.0-Based Extension//EN"
 "http://www.FreeBSD.org/XML/share/xml/freebsd50.dtd">

<book xmlns="http://docbook.org/ns/docbook"
  xmlns:xlink="http://www.w3.org/1999/xlink" version="5.0"
  xml:lang="en">

  <info>
    <title>An Example Book</title>

    <author>
      <personname>
        <firstname>Your first name</firstname>
        <surname>Your surname</surname>
      </personname>

      <affiliation>
 <address>
   <email>foo@example.com</email>
 </address>
      </affiliation>
    </author>

    <copyright>
      <year>2000</year>
      <holder>Copyright string here</holder>
    </copyright>

    <abstract>
      <para>If your book has an abstract then it should go here.</para>
    </abstract>
  </info>

  <preface>
    <title>Preface</title>

    <para>Your book may have a preface, in which case it should be placed
      here.</para>
  </preface>

  <chapter>
    <title>My First Chapter</title>

    <para>This is the first chapter in my book.</para>

    <sect1>
      <title>My First Section</title>

      <para>This is the first section in my book.</para>
```

```
    </sect1>
  </chapter>
</book>
```

A.2. `Artigo` DocBook

Exemplo A.2. `Artigo` DocBook

```
<!DOCTYPE article PUBLIC "-//FreeBSD//DTD DocBook XML V5.0-Based Extension//EN"
 "http://www.FreeBSD.org/XML/share/xml/freebsd50.dtd">

<article xmlns="http://docbook.org/ns/docbook"
  xmlns:xlink="http://www.w3.org/1999/xlink" version="5.0"
  xml:lang="en">

  <info>
    <title>An Example Article</title>

    <author>
      <personname>
        <firstname>Your first name</firstname>
        <surname>Your surname</surname>
      </personname>

      <affiliation>
 <address>
   <email>foo@example.com</email>
 </address>
      </affiliation>
    </author>

    <copyright>
      <year>2000</year>
      <holder>Copyright string here</holder>
    </copyright>

    <abstract>
      <para>If your article has an abstract then it should go here.</para>
    </abstract>
  </info>

  <sect1>
    <title>My First Section</title>

    <para>This is the first section in my article.</para>

    <sect2>
      <title>My First Sub-Section</title>

      <para>This is the first sub-section in my article.</para>
    </sect2>
  </sect1>
</article>
```

Índice Remissivo